Box 4
folder 10

3 of 4

102739322

≡ Your Turn ≡

How about it? How are we doing? This is the place to send us your message. Tell us what you like and what you don't, what you'd like to see but don't — or whatever other feedback you'd like to send. Send comments to Marie A. Lenzi, 225 Rector Place, NY, NY 10280; fax to THE HOTLINE, *(212)274-0646; or, if you're ordering, send along with your order.*

Please check whichever box applies:

	Read entirely	Scanned	Found helpful	Will refer to	Didn't read
Feature	❑	❑	❑	❑	❑
Languages	❑	❑	❑	❑	❑
Getting Started	❑	❑	❑	❑	❑
Statistics	❑	❑	❑	❑	❑
Silver Bullets	❑	❑	❑	❑	❑
Product Review	❑	❑	❑	❑	❑
FYI	❑	❑	❑	❑	❑
Product Announcements	❑	❑	❑	❑	❑
Calendar	❑	❑	❑	❑	❑

Please rate the following ***1*** (dislike or strongly disagree) to ***10*** (like or strongly agree):

Presentation of material __ Comments: _____

Accuracy of material __ Comments: _____

Overall helpfulness in your job __ Comments: _____

What else would you like to see in THE HOTLINE? _____

Any other comments? _____

Name, title & phone (optional): _____

hotline on
OBJECT-ORIENTED
technology

❑ ***Yes,*** plug me into the latest thinking and developments in object-oriented technology. Enter me as a subscriber at the term marked below and rush me the current issue. This is a risk-free offer — I may cancel my subscription at any time and promptly receive a refund for the unused portion.

1 year (12 issues) ❑ $249 *(outside US add $30 per year for air service)*

2 years (24 issues) ❑ $478 *(save $20)*

Back issues @ $25 each *($27.50 foreign)*:
Vol.1, Nos. _____
Vol.2, Nos. _____

❑ **Phone order**
Call Subscriber Services
(212)274-0640

❑ **Fax order**
Fill out term and name/address information, then fax to:
(212)274-0646

❑ **Send me a copy of**
THE INTERNATIONAL OOP DIRECTORY
@ $63 ($71 foreign)
(all directory orders must be prepaid; fax or mail credit card information to SIGS, 588 Broadway, #604, NYC 10012; make check payable to OOP Directory. Foreign orders must be prepaid in US dollars drawn on US bank.) D1JC

❑ **Bill me**
❑ **Check enclosed**
❑ **Credit card orders**
❑ MasterCard
❑ Visa

Card# Expiration Date

Signature

Name

Company

Street/Building#

City/Province

ST/Zip/Country

Telephone D1JC

Mail (and make out check, if applicable) to:
THE HOTLINE
Subscriber Services
P.O. Box 3000, Dept. HOT
Denville, NJ 07834
(foreign orders must be prepaid in US dollars drawn on a US bank)

Send me a complimentary copy of your related publication(s):

❑ THE JOURNAL OF OBJECT-ORIENTED PROGRAMMING

❑ OBJECT MAGAZINE

❑ THE C++ REPORT

Call Subscriber Services
(212)274-0640
or send order to
SIGS, 588 Broadway, #604
New York, NY 10012

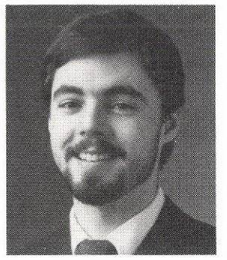
D1JC

hotline on
OBJECT-ORIENTED
technology

VOLUME 2, NUMBER 11 THE MANAGER'S SOURCE FOR TRENDS, ISSUES & STRATEGIES SEPTEMBER 1991

From applications to frameworks

One of the battle cries of object-oriented programming is "why reinvent the window?"



Zack Urlocker

WHY REINVENT THE APPLICATION?

While object-oriented programming is very effective in eliminating the need to reinvent user interface objects, the same abstraction techniques can be used to supply even more generic application components. When combined together with a flexible, open architecture, these components form what is known as an application framework.

You can think of an application framework as the basic foundation upon which all applications are built. Most likely, any significant application you've ever written contains routines that you've reused from previous programs to supply command processing, windowing, menus, data management, and so on. However, tieing these different pieces together is often more difficult in practice than in theory since each library may depend on different, sometimes incompatible, assumptions.

An application framework ties reusable components into a generic structure so that they can be easily inherited into other applications.

An application framework is not just a library of objects, it is also the underlying architecture that accommodates the generic components. Thus, an application framework is like having the basic structure of a house, including electrical wiring, central air conditioning, and heating, plus a warehouse of furniture, doors, fixtures, and windows (of course!). Since the application framework provides only generic components, you can think of it as a fully functioning application that does nothing, but can be extended to do anything. The way you extend an application framework is by adding new objects.

In this article, I will describe the evolution of application frameworks including Smalltalk's Model-View-Controller (MVC) paradigm and MacApp. I will then provide an examination of two application frameworks developed by Borland International: Turbo Vision for MS-DOS and ObjectWindows for Microsoft Windows. These frameworks build on many of the ideas established in MVC and MacApp and provide PC application de-

continued on page 3 ...

IN THIS ISSUE

FEATURE — From applications to frameworks *Zack Urlocker*	1	2	FROM THE EDITOR
LANGUAGES — Report on the seventh meeting of the Object-Oriented COBOL Task Group *Adams & Lenkov*	6	18	PRODUCT REVIEW — APPLICATION ORGANIZER PLUS VERSION 1.0 *reviewed by David Bellin*
GETTING STARTED — Getting started with object technology: effectively planning for change *Mark Lorenz*	9	19	FYI — Industry publications excerpts
STATISTICS — Object statistics on the way *Marie A. Lenzi*	13	24	PRODUCT ANNOUNCEMENTS/NEW PARTNERINGS & ACQUISITIONS
SILVER BULLETS AND WEREWOLVES — On objects and bullets *Adnan Adam Onart*	16	27	CALENDAR

From the Editor



Among the buzz words of the object technology subculture is that of application frameworks. This is a term that, like most object terms, is not well defined and of course abused. What precisely are application frameworks? How do they differ from applications? How do they support the object application development? What are some examples of application frameworks? In "From Applications to Frameworks," Zack Urlocker of Borland provides a clear description of application frameworks and their place in object application development. The most notable aspect of our feature article is that application frameworks are part of the infrastructure of your organization's class hierarchy—targeted sets of components designed to easily support inheritance from user applications.

Learning object technology is not a monumental undertaking; neither is it a trivial one. Changing your way of thinking about the problem/solution domain, coupled with changes in your day-to-day work practices, suggest a fairly sizable adjustment. Not an impossible transition to make; however, patience, perseverance, tenacity, and support from all sides is imperative. Mark Lorenz of IBM Cary Software Technology Center offers a perspective on making object technology your own. "Getting Started with Object Technology: Effectively Planning for Change" draws a parallel between object software development and the old craft guilds. Coupled with approximate time frames, this perspective is valuable in ascertaining when levels of expertise are achieved and, therefore, when major advancements in productivity can be expected.

The largest impediment to software quality and development productivity is not new technology but the environment in which the technology is made to exist. The technology itself, whether it be objects, AI, or CASE, is comparatively easy. The difficult part is the sociopolitical/cultural issues surrounding the implementation of the new technology. Not notorious as an industry for our interpersonal and communication skills, we are faced with at least 2.5 strikes against us before we step up to the paradigm shift plate. This impediment to forward progress frustrates many, myself included. However, Adnan Onart of Timeplex has taken on the challenge. As the "Turkish werewolf hunter in America," he believes that "objects are the silver bullets." Adnan is vigilant and determined to improve the working environment. He has quite a challenge ahead of him. I know—we are sharing a supply of silver bullets.

As object COBOL gains more and more press, and as products begin to ooze into the marketplace, the promise of significant improvements to this widely accepted language threatens to become a reality. Megan Adams and Dmitry Lenkov of the Hewlett-Packard California Language Laboratory report on the current progress of the Object COBOL Task Group and what object COBOL means to organizations participating in this massive technology investment.

David Bellin offers a first look at Application Organizer Plus version 1.0, a set of tools for developers working in an Objectworks/Smalltalk environment.

And, finally, a bit on object technology statistics, well almost. Recently, I came across a flavorless study that offered a magnificent launching platform for evaluation work in the area of objects and reusability. The study started out to support CASE technology and ended up establishing start-up parameters and requirements for future studies. Maybe we will finally discover that it is not the tools, but the concepts supported by the tools that improve productivity. Even though there is currently nothing particularly solid, "Object Statistics are on the Way."



Editor

Marie A. Lenzi

SIGS Publications, Inc. Advisory Board

Tom Atwood, *Object Design*
Grady Booch, *Rational*
George Bosworth, *Digitalk*
Brad Cox, *Information Age Consulting*
Chuck Duff, *The Whitewater Group*
Adele Goldberg, *ParcPlace Systems*
Meilir Page-Jones, *Wayland Systems*
Tom Love, *Consultant*
Bertrand Meyer, *Interactive Software Engineering*
P. Michael Seashols, *Versant Object Technology*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

HOTLINE Editorial Board

Jim Anderson, *Digitalk, Inc.*
Larry Constantine, *Consultant*
Mary E.S. Loomis, *Versant Object Technology*
Reed Phillips, *Knowledge Systems Corp.*
Trygve Reenskaug, *Taskon A/S*
Zack Urlocker, *Borland International*
Steven Weiss, *Wayland Systems*

SIGS Publications, Inc.

Richard P. Friedman, Founder & Group Publisher

Art/Production

Elisa Varian, Production Manager
Susan Culligan, Creative Director
Kristin R. Juba, Production Editor
Caren Polner, Desktop Designer

Circulation

Diane Badway, Circulation Business Manager
Kathleen Canning, Fulfillment Manager
John Schreiber, Circulation Assistant

Administration

David Chatterpaul, Bookkeeper
Suzanne W. Dinnerstein, Conference Manager
Jennifer Fischer, Assistant to the Publisher
Laura Lea Taylor, Administrative Assistant

Margherita R. Monck, General Manager

hotline on OBJECT-ORIENTED *technology*

THE HOTLINE ON OBJECT-ORIENTED TECHNOLOGY (ISSN #1044-4319) is published monthly by SIGS Publications, Inc., 588 Broadway, NY, NY 10012, (212)274-0640.

© Copyright 1991 SIGS Publications, Inc. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the U.S. Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher. Mailed First Class. Subscription rate — one year (12 issues) $249, Foreign and Canada $279. Single copy $25.

POSTMASTER: Send address changes & subscription orders to The HOTLINE, Subscriber Services, P.O. Box 3000, Dept HOT, Denville, NJ 07834.

Submit editorial correspondence to Marie A. Lenzi, 225 Rector Place, Suite 12N, New York, NY 10280. voice: (212)945-9447, fax: (212)786-4032.

SIGS PUBLICATIONS

Publishers of *Hotline on Object-Oriented Technology, Journal of Object-Oriented Programming, Object Magazine, The X Journal, The C++ Report, The Smalltalk Report,* and *The International OOP Directory.*

Back issues available

Vol.2, No.10/Aug. '91 ≡ Distributed object management: improving worker productivity • *Osher* ≡ Getting the best from objects: the experience of HP • *Coleman & Hayes* ≡ APPLICATIONS: EC employs object technology ...• *Stein* ≡ CAPACITY PLANNING: Fiddling while ROMs burn • *Rovira*

Vol.2, No. 9/July '91 ≡ Multimedia is everywhere! • Weiss & Berman ≡ Developing an object technology prototype • Kulinek ≡ Object-oriented capacity planning • Charles-A. Rovira.≡ How OOP has changed our developmental lifecycle • Zeik≡ Modularization of the computer system • Per-Anders Angenius

Vol.2, No. 8/June '91 ≡ Domain of objects: the Object Request Broker • *Dyson* ≡ Object-based approach to user documentation • *Reeder* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Do we need object-oriented design metrics? • *Hopkins*

Vol.2, No.7/May '91 ≡ Hybrid object-oriented/functional decomposition for software engineering • *Henderson-Sellers* ≡ So, what makes object databases different? (Part 4) • *Blakey* ≡ Using the generic application to solve similar domain problems • *Gossain* ≡ Experiences using CLOS • *Hopkins* ≡ International Conference on Object-Oriented Technology, Singapore • *Bennett*

Vol.2, No.6/Apr. '91 ≡ An artist's perspective of programming • *Rovira* ≡ So, what makes object databases different? (Part 3) • *Blakey* ≡ Moving from Pascal to C++, Part 3 • *Gole* ≡ Object projects: what can go wrong • *Stewart* ≡ Reflections from LOOK-'91 • *Anders-Angenius*

Vol.2, No.5/Mar. '91 ≡ Large-scale object systems development • *Morgan & Davis* ≡ Data definition languages • *Blakey* ≡ Object disorientation • *Kerr* ≡ Moving from Pascal to C++, Part 2 • *Keller* ≡ The Object Management Group • *Barber*

Vol.2, No.4/Feb. '91 ≡ Moving from Pascal to C++ • *Sollisch* ≡ The Australian object-oriented scene • *Henderson Sellers* ≡ So, what makes object databases different? • *Blakey* ≡ Managing the evolution • *Novobilski* ≡ 1990: the year of realism • *Love* ≡ Choosing a configuration for object development • *Moody*

Vol.2, No.3/Jan. '91 SPECIAL ISSUE: CASE AND METHODOLOGY ≡ Object orientation: stop, look and listen! • *Page-Jones* ≡ The spiral model for object software development • *Wasserman & Pircher* ≡ Bootstrapping object-oriented CASE • *Wybolt* ≡ A CASE tool for object software development • *Ackroyd* ≡ Comments on "Race for the database pennant" • *Loomis*

Vol.2, No.2/Dec. '90 ≡ Waiting for OODB2 • *Durham* ≡ Object modeling • *Singh* ≡ Object-oriented COBOL — the next generation • *Adams & Lenkov* ≡ Object-oriented training in the UK • *Hopkins*

Vol.2, No.1/Nov. '90 ≡ Objects by teamwork • *Constantine* ≡ Race for the database pennant: a third-generation manifesto • *Stein* ≡ Project team descriptions *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.12/Oct. '90 ≡ Crown jewels: the value of OOT in MIS departments • *Bonar* ≡ Why is an object database faster than a relational system? • *Loomis* ≡ Visual programming object language environments • *Klausner* ≡ Hype — necessarily (or a necessary) evil? • *Adams* ≡ ON MANAGING: Déjà Vu • *Lenzi*

Vol.1, No.11/Sep. '90 ≡ Introducing OOPS through design — it's all in the words • *Copen Goldstein* ≡ The Australian object-oriented scene • *Henderson-Sellers* ≡ Objects on tour (Review of Borland's OOP World Tour) • *Lenzi* ≡ Design principles for modular product development • *Bennett* ≡ Asset management and object-oriented technology • *Hopkins & Warboys*

Vol.1, No.10/Aug. '90 ≡ WyCASH+: An application built within an OOP environment • *McCullough & Deshler* ≡ Mathematics and methodology • *Martin* ≡ Organizing for reuse • *Goldberg & Rubin* ≡ CALS Expo–89 Conference • *Purdy*

Vol.1, No.9/July '90 ≡ Cognos and Eiffel: a cautionary tale • *Leathers* ≡ What's in an object anyway? • *Novobilski* ≡ The impact of Windows 3.0 on the OOP market • *Urlocker* ≡ Introducing reuse into the development process • *Goldberg & Rubin*

Vol.1, No.8/June '90 ≡ A taste of the USENIX C++ Conference • *Wells* ≡ Comparing relational and object database management systems part 2 — Composites, flexibility, and extensibility • *Martin* ≡ Case studies in prototyping, part 3 • *Goldberg & Rubin* ≡ Object-oriented requirements analysis • *Berard*

Vol.1, No.7/May '90 ≡ Object-oriented development: not business as usual • *Lorenz* ≡ Comparing relational and object database management systems part 1 — Single paradigm programming is the key difference • *Martin* ≡ Case studies in prototyping, part 2 • *Goldberg & Rubin* ≡ Understanding the recursive/parallel lifecycle • *Berard*

Vol.1, No.6/April '90 ≡ Lifecycle approaches • *Berard* ≡ A data manipulation language for C++ developers — a proposal • *Atwood* ≡ Case studies in prototyping, part 1 • *Goldberg & Rubin* ≡ I already learned in kindergarten all I needed to know about object-oriented design • *Ramakrishnan*

Vol.1, No.5/March '90 ≡ Objects of desire • *Stone* ≡ OMG: The state of the object-oriented DBMS market • *Atwood* ≡ Talking to project managers • *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.4/Feb. '90 ≡ Greed: a complete example • *Love* ≡ OMG: an industry push for standards • *Atwood* ≡ Talking to project managers • *Goldberg & Rubin* ≡ OOP in the 1990s • *Urlocker*

Vol.1, No.3/Jan. '90 ≡ Adopting object-oriented techniques: management and organizational impacts • *Lenzi* ≡ The object-oriented database system manifesto: a consensus from academia • *Atwood* ≡ Which language to choose, revisited • *McKenna*: On language wars • *Stroustrup*

To order, see back page; for reprints of individual articles, call Michael Biggerstaff at Reprint Management Services (215)286-7017.

Calendar

THE HOTLINE CALENDAR presents conferences and meetings that focus exclusively on object-oriented technology. To have a meeting or conference listed, please send the dates, conference name and location, sponsor(s) and contact name and telephone number to the Editor: Marie A. Lenzi, 225 Rector Place, Suite 12N, New York, NY 10280.

October 6–11 OOPSLA	October 28–31 International Working Conference on the Object-Oriented Approach in Information Systems	October 28–31 SCOOP-Europe	November 18–22 C++ at Work
Phoenix, AZ	Quebec City, Canada	Westminister, London, UK	Santa Clara, CA
Sponsor: ACM	Sponsor: IFIP	Sponsor: *JOOP*, *Object Magazine*, & Wang Institute of Boston University	Sponsor: *The C++ Report* & Wang Institute of Boston University
Contact: Timlynn Babitsky, (803) 957-0648	Contact: Bernard Moulin, Fax (418) 656-2324	Contact: Suzanne W. Dinnerstein, (212) 274-0640	Contact: Suzanne W. Dinnerstein, (212) 274-0640

New Partnerings and Acquisitions, September 1991

Borland International, Inc. agreed to purchase **Ashton-Tate Corp.** Completion of the deal is subject to government approval and approval by the company's shareholders. The deal is expected to close by October.

Microsoft Corp. announced the promotion of Jim Allchin to vice president of advanced Windows services. Allchin will head a group developing information access, object orientation, and distributed processing capabilities for Windows. Paul Leach will join Allchin's team as director of future systems. Leach was a co-founder of Apollo Computer, Inc. and most recently was a fellow at Hewlett-Packard/Apollo. Allchin left Banyan Systems, Inc. last December to become a manager with Microsoft's Network Business Unit.

Digital Equipment Corp. Europe and Information Systems Experts, Inc. formed a partnership to develop and market software that integrates sales, marketing, and customer services functions. ISE's involvement will cut the software development time by at least 50% using rapid prototyping and object-oriented programming. ISE has written the design specification based on five years of market research on sales and marketing automation.

AICorp, Inc. announced that **Carnegie Group** has joined its Strategic Alliance Program and will provide consulting and systems development services to users of AICorp's KBMS, the Knowledge Base Management System. AICorp will also license and resell Carnegie Group's ROCK, the Representation of Corporate Knowledge, as part of the KBMS toolkit. ROCK, an implementation of the Initiative for Managing Knowledge Assets technology, is a foundation technology for developing and deploying large-scale and complex knowledge base systems.

ParcPlace Systems' Objectworks/Smalltalk and **Versant Object Technology's** object-oriented database are to become available on **Sequent Computer Systems'** Symmetry 2000 systems following an agreement between the three.

ParcPlace Systems and **Gemini Consulting** announced a corporate investment and software development agreement. Under the terms of the agreement, Gemini Consulting, a subsidiary of the Paris-based **Sogeti Group,** has made an equity investment in ParcPlace. Thomas Madison, Senior Vice President of Gemini Consulting, has been elected to the ParcPlace Board of Directors. As part of the agreement, the two companies will work together to enhance object-oriented technology to further address the need of Gemini Consulting's large base of corporate clients.

ParcPlace Systems announced that it has signed a distribution agreement and begun shipment of FACETS, a fourth generation language application development tool from **Reusable Solutions** to be used with Objectworks/Smalltalk. The combination of FACETS and Objectworks/Smalltalk delivers an extendable object-oriented 4GL development environment by providing the capabilities of traditional 4GLs and a migration path to object-oriented technology.

Servio Corp., developer of the GemStone object-database system, announced object database access for Macintosh C applications with support of **Symantec Corp.'s** THINK C software development environment. Servio says new and existing Mac client applications written in Symantec's THINK C can now access objects stored in the GemStone object server. THINK C is Symantec's C programming language for the Mac, implemented with object-oriented extensions. GemStone is a client/server object-database management system with interfaces to support Smalltalk, C, and C++ applications.

Saber Software, Inc. announced a major purchase agreement under which it will provide 300 licenses of its Saber-C++ and 100 licenses of its Saber-C programming environments to SunSoft, the systems software subsidiary of **Sun Microsystems,** Inc. The products will be used for both ongoing and new development projects.

McDonnel Douglas Systems Integration Co. will market and sell **Hansen Software's** line of municipal infrastructure management applications. Hansen products, primarily utility and street management systems, enhance the capabilities of the McDonnell Douglas Graphics Design Systems by providing cohesive, ready-to-use applications. Built around a relational database, the software complements GDS's object-oriented graphics database.

KnowledgeWare, Inc. and **IntelliCorp, Inc.** confirmed discussions on a possible merger or other business relationship. Industry observers have suggested that KnowledgeWare's primary interest in IntelliCorp was in acquiring the company's expertise in object-oriented, client/server-based development technology.

IBM announced its first SystemView development partners: **Bachman Information Systems, Inc., Candle Corp., Goal Systems International, Inc.,** and **Platinum Technology, Inc.** As development partners, each of the firms will participate in SystemView design councils with IBM and users.

IBM announced that it extended its relationship with **Borland International** for marketing rights to Borland's ObjectVision object-oriented application development tools. The new agreement gives IBM nonexclusive marketing rights to the new version of ObjectVision, which works with OS/2 2.0, due later this year.

IBM announced that it would resell **Lotus'** cc:Mail and Notes as part of OfficeVision. IBM officials have confirmed that Microsoft's Network Courier is off the sell list.

Mike Taylor has been named the new president of **Instantiations.**

... continued from page 1

velopers with a head start in creating sophisticated applications.

IN THE BEGINNING: MVC

Like object-oriented programming itself, application frameworks are not new. The earliest application framework was developed at Xerox's Palo Alto Research Center in conjunction with the Smalltalk-80 object-oriented development system. This system was known as the Model-View-Controller system or MVC for short. Although MVC and its variants are more of a programming paradigm than a complete framework, MVC is an important starting point for understanding the evolution of application frameworks.

The basic notion of MVC is that all applications can be broken down into three components: the Model is the data that is being analyzed, the View provides a visual, on-screen representation of the data, and the Controller provides the end user with a mechanism for effecting the Model and, hence, the View. For example, if we consider a simple business graphics program, the Model is the underlying data being charted (perhaps a two-dimensional array of strings and numbers), the View is the chart on screen, and the Controller is what allows the user to click with the mouse and edit the underlying data.

An application usually consists of many Views. For example, a chart might be a part of a more complex display with a list box and a text window. With MVC, there is always a separate Controller for each View. Thus, our more complex application would have at least three Views, each with a corresponding Controller that acts as a message dispatcher to respond to user input in the View.

The View and Controller are always linked together. Whenever the Model changes, it will send a message to all dependent Views so that they can update the visual representations. Thus, if the user clicked on the chart to edit the data, a message would be sent to each of the Views, and they would receive the most recent data from the Model. This message-based system for updating Views has influenced the Macintosh, Windows, and other graphical environments.

The MVC system provides generic classes for the View and Controller. There is no generic class for the Model since the data can be in any format. Smalltalk-80 provides many reusable types of Views and Controllers, thereby giving a standard set of user-interface components. MVC may look lean but, since it's based on Smalltalk, there is a large library of data management classes already included.

Although MVC was an early step in the right direction, it has suffered from some limitations. The lack of documentation has made MVC difficult to learn, and MVC programs tend to be difficult to modify. A more fundamental problem is that the separation of View and Controller does not fit well with the event-driven architecture that is now supplied in most graphical user interfaces including the Macintosh, Microsoft Windows, and OS/2 Presentation Manager. On these systems, the operating environment automatically manages the event process, passing user events to the program or in some cases to the window itself. For this reason alone, many Smalltalk users consider MVC to be ancient history.

MACAPP: A SECOND GENERATION FRAMEWORK

In 1984, many programmers who examined the Macintosh were overwhelmed with the richness of the toolbox ROMs and built-in routines for windowing, menus, mouse support, quickdraw graphics, etc. However, even more programmers were overwhelmed with the complexity of programming for the Mac. It seemed as if every programmer was destined to write reams of standard code just to put up a window on the screen. With this problem in mind, Apple Computer created MacApp, an extensible application toolkit or, in other words, an application framework.

MacApp is a library of object-oriented routines that supply much of the standard code required of every Macintosh application. MacApp includes support for windows, dialogs, input fields, buttons, command processing, and printing. MacApp is written in Object Pascal and is accessible from several languages including both Object Pascal and C++ under the Macintosh Programmer's Workbench (MPW).

Since MacApp is based on the Macintosh system that includes an event-driven architecture, MacApp eliminates the separate use of a Controller as found in MVC in favor of a simpler two-part model based primarily on Data and Views.

MacApp includes a hierarchy of object types such as TApplication, TView, TScroller, TList, etc., that provide the basic capabilities of all Macintosh applications. MacApp has been used to create a variety of commercial applications including the MacCalc spreadsheet and MacTerminal communications program. Although MacApp impacts on both speed and space, for most programmers MacApp saves precious programming time and is worth the cost.

TURBO VISION FOR MS-DOS

Shortly after Borland released Turbo Pascal with Objects, we began work on an application framework for MS-DOS. It is similar in some ways to MacApp, though it operates in text mode on the IBM PC and compatibles.

Turbo Vision supplies a framework of classes or object types and methods contained in various precompiled units. These range from the thoroughly generic TAp-

plication and TView types, similar to those found in MacApp, to mini-applications such as TCalculator, TCalendar, and TEditor, a full-blown WordStar-compatible text editor with pull-down menus and mouse support. It also includes generic data management types similar to those found in Smalltalk for creating collections and persistent objects stored on disk via streams. Many of its object types are abstract types for which users will normally create descendants. However, these abstract types serve two purposes: they group common code that would otherwise have to be written by every programmer and they define a protocol for descendant types. Abstract types are the key to having a generic application that can be easily customized. It is the reason we can define generic capabilities that will be implemented in descendant types implemented by the user of the framework. For example, all descendants of the TView type are expected to respond to a draw message when the user uncovers the object on the screen. The protocol specifies that they will receive and respond to a draw message, but it does not specify how they respond to the message. That action is implemented at the application level.

OBJECTWINDOWS FOR MS-WINDOWS

Windows programmers have an even more daunting task. After all, with an API of over 500 functions, Windows practically begs to have an application framework to simplify the job.

The ideal situation would have been to allow programmers to simply recompile Turbo Vision programs and miraculously create a Windows application. Although this is possible in theory, it would have either added tremendously to the complexity of Turbo Vision or would have made too many limiting assumptions for Windows applications.

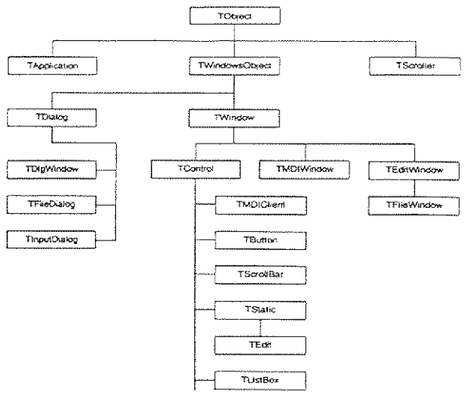


Figure 1: Diagram of ObjectWindows hierarchy.

Instead, we implemented a similar, though not identical, application framework for Windows called ObjectWindows. Architecturally, the two frameworks are very similar. Both provide an event-driven architecture and have many common classes. For example, the data management classes are identical. The fundamental difference is that while Turbo Vision is a text mode framework built on DOS, ObjectWindows runs exclusively in graphics on Windows and uses many of the built-in capabilities of Windows for message dispatching and user-interface management. Each framework is optimized for its native environment, giving full access to all underlying facilities.

USING AN APPLICATION FRAMEWORK

Although you can use portions of the application framework hierarchy, such as the collections and streams, in existing applications without using the entire framework, an application framework is much better used as a starting point for new applications. The starting point for a program is the generic TApplication type. All applications inherit the event handling and mouse support from this generic type. Thus, the simplest application would simply inherit from TApplication and add new methods to supply nongeneric capabilities. The main program therefore consists of only three lines: initialize the application, run the event loop, and exit when finished.

Listing 1 shows a typical "Hello, world" application for Windows that uses the Pascal version of the ObjectWindows application framework.

DATA MANAGEMENT

Most applications spend a lot of time creating and manipulating fairly standard data structures: dynamic arrays, linked lists, hash tables, etc. Although this code is not particularly difficult, there is not much to be gained by rewriting it for every application. Turbo Vision and ObjectWindows share a common set of generic data management types including collections and streams.

The generic TCollection type and its descendants provide the equivalent of dynamic arrays and linked lists. A collection can contain different types of objects, though typically it will contain descendants of an ancestor type. You can easily search, sort, add, and delete items in a collection. You can also perform operations on all items in the collection. For example, you might want to iterate through a collection of window objects and get the location of each one. That would be accomplished by sending a forEach message to the collection and supplying a procedure parameter. Collections also provide the ability to create persistent objects stored on disk. Though these persistent objects are not part of an object-oriented

BLOCS
Berkeley Productivity Group
35032 Maidstone Court
Newark, CA 94560
(415) 795-6086

Berkeley Productivity Group (BPG) announced a commercial release of **BLOCS,** the Berkeley Library of Objects for shop floor Control and Scheduling. BLOCS is a suite of open applications based on reusable objects that represent manufacturing entities and concepts such as lot, machine, and work station. These objects are assembled into applications that meet generic manufacturing requirements, but they can also easily be extended to meet the unique needs of a particular manufacturing facility.

ONTOS Release 2.1
ONTOS, Inc.
Three Burlington Woods
Burlington, MA 01803
(617) 272-7110

ONTOS, Inc. announced **ONTOS Release 2.1**, an object database management system for UNIX. This release was designed to address the needs of the growing number of ONTOS customers ready to deploy distributed applications such as network management and data integration systems. The goal of Release 2.1 was to enhance the quality and offer the first production-oriented, industrial-strength object database.

ObjectGraphics
The Whitewater Group
1800 Ridge Ave.
Evanston, IL 60201
(708) 328-3800

Whitewater announced the newest version of **ObjectGraphics,** a high-level, object-oriented graphics library that extends the power of ObjectWindows in Turbo Pascal for Windows, Borland C++, and Actor to create comprehensive, platform-independent support for graphics, including diverse shapes, bitmaps, icons, multiple fonts, and text styles. ObjectGraphics completely masks the graphics engine in Windows and allows users to develop graphical applications using a simple set of graphic objects rather than using scores of primitive procedure calls.

Saber-C Version 3.1.
Saber Software, Inc.
10 Fawcett St. Cambridge,
MA 02138-1110
(617) 498-3000

Saber Software, Inc. announced the release of **Saber-C Version 3.1.** New features include: preprocessor support, including embedded SQL preprocessors of Oracle, INFORMIX, and other industry standard SQL databases; support of dynamic shared libraries on Sun platforms; support for HP 9000 Series 400 workstations; and support for loading FORTRAN object code.

DOS/4G
Rational Systems, Inc.
220 N. Main St.
Natick, MA 01760
(508) 653-6006

Rational Systems, Inc. announced **DOS/4G,** a 32-bit extender that complies with the DOS Protected Mode Interface and is available without costly royalty fees. DOS/4G allows DOS extended applications to run as a DOS program under any DPMI-compliant operating system such as Microsoft Windows 3.0/3.1. DOS/4G takes advantage of 80386/486-based Intel architectures by allowing users to run 32-bit applications in protected mode. DOS/4G has a small memory footprint — less than 25KB in real mode — that leaves considerable memory for user applications. Price ranges from $5,000 to $25,000, depending on system configuration.

Rational Consulting announced that it is distributing and supporting **The C++ Booch Components,** a reusable software component library. The Booch Components are available on a variety of platforms, including IBM PC compatibles, Macintoshes, and UNIX workstations as well as mini computers and mainframe systems. The C++ Booch Components represent the second generation of a widely used and mature component library, the Ada Booch Components. The C++ components can be used with any C++ compiler that is compatible with AT&T's C++ version 3.0. A template preprocessor is available for version 2.0 compatibility. Rational Consulting is a division of Rational.

IntergrAda/386
AETECH
380 Stevens Ave., Ste. 212
Solana Beach, CA 92075
(619) 755-1277

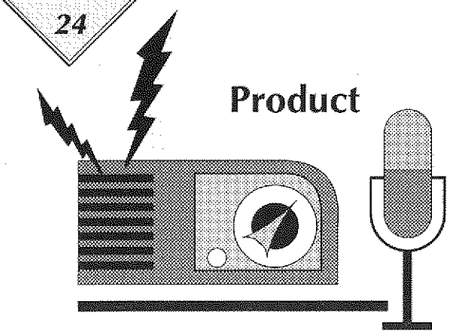
AETECH announced the release of **IntergrAda/386.** IntegrAda/386 is an integrated Ada Programing Support Environment (ASPE) with a production-quality compiler that has been fully validated by the government to the latest Ada Compiler Validation Capability. IntergrAda, Ver 4.20 was chosen by the U.S. Air Force as the standard military MS-DOS PC Ada compiler and tools for the DESKTOP III contract. IntegrAda/386 includes the same unique Ada design and programming features in an environment that encapsulates the tools, compiler, and libraries so that the programmer never has to leave the programming environment. The seamless environment is modeled after C and Pascal Programming systems.

ANDOR ONE
Hydra Systems, Inc.
1340 S. Saratoga-Sunnyvale
Rd., Ste. 106
San Jose, CA 95129
(408) 253-5800

Hydra Systems, Inc. announced the delivery of its **ANDOR ONE** product, a PC add-in board with associated software that brings the functionality of Apple's Macintosh to the PC world. A full-length PC card installs into any PC ranging from the earliest PC-XTs through today's high-performance 486 machines. Through ANDOR ONE, many of the PC's peripherals such as the keyboard, mouse, video display, hard disk, and floppy disk drive can be accessed from Macintosh Software.

Software Development Kit
NetManage, Inc.
10020 N. DeAnza Blvd., #101
Cupertino, CA 95014
(408) 257-6404

NetManage announced the release of a comprehensive **Software Development Kit** for building distributed applications using ONC RPC/XDR. Using NetManage's RPC/SDK, Windows 3.0 applications can now interact and exchange data with any other operating system or hardware platform that supports ONC RPC/XDR including IBM, DEC, Sun, HP, DOS, OS/2 NetWare, SCO, etc. The RPC/SDK implements the complete Sun release 4.0 ONC Remote Procedural Call (RPC) and the External Data Representation (XDR) mechanism (except DES authentication). As a Windows Dynamic Link Library (DLL) the RPC DLL is fully reentrant, multitasking, and does not require any of the PC's 640K base memory.



Product Announcements

Product Announcements is a service to the readers of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY; it is neither a recommendation nor an endorsement of any product discussed.

Telekurs Database Server

Telekurs
11 Broadway
New York, NY 10004
(212) 487-2700

Telekurs North America, a subsidiary of Telekurs AG of Zurich, Switzerland, announced the U.S. introduction of the **Telekurs Database Server for the UNIX environment.** The system creates and maintains a flexible, object-oriented database of realtime and fundamental financial information delivered by the Telekurs Ticker Service.

GTX Object Repository

Object Databases
238 Broadway
Cambridge, MA 02139
(617) 354-4220

Object Databases announced the release of the **GTX Object Repository.** GTX supports large multimedia databases consisting of complex, linked data types with image, voice, and video objects; fault tolerant network applications requiring a strong transaction model and detailed audit trails; real-time, high volume data capture with a requirement for immediate query capability; and recall of temporal object versions required for group work and on-line back-up.

CodeBase++ 1.04

Sequiter Software, Inc.
Ste. 209, 9644 - 54 Ave.
Edmonton, Alberta T6E 5V1
Canada (403) 448-0313

Sequiter Software announced the release of **CodeBase++ 1.04,** a C++ class library for database management that now includes support for the Clipper .NTX index files. Multiple NDX/NTX index files are opened automatically when a datafile is opened thus insuring that all necessary files are immediately accessible. The similarity between this new implementation and the production .MDX indexes means that developers need not write different code for different indexes, but need only recompile. CodeBase++ retails for U.S. $295 and comes with a prebuilt DLL for Microsoft Windows 3.0, 100% of the source code, and a 60-day money-back guarantee.

Prograph version 2.5

TGS Systems
2745 Dutch Village Rd.,
Ste. 200 Halifax, Nova
Scotia Canada
(902) 455-4446

TGS Systems introduced version **2.5** of **Prograph,** its object-oriented Visual Programming Environment for the Macintosh. In addition to adding a wide array of new features to the Prograph environment, this new version provides high-level System 7.0/IAC support and a database engine. Prograph will also connect to SQL databases through interfaces for DAL and Oracle. These interfaces are part of the company's new line of add-on products.

Greenleaf Comm++

Greenleaf Software, Inc.
16479 Dallas Pkwy.,
Ste. 570 Dallas, TX 75248
(214) 248-2561
(800) 523-9830

Greenleaf Software, Inc. released **Greenleaf Comm++,** a class library for asynchronous communications. As a C++ library, it provides a hierarchy of classes that give the programmer simple access and control of serial communications with or without terminal emulation.Greenleaf Comm++ was designed with the intention of making the use of serial asynchronous communications within a C++ program as natural to the user as possible. The communication related classes included with the library were designed to overcome the limitations of addressing the serial RS-232 ports through the operating system or BIOS. Greenleaf Comm++ supports the IBMPC, XT, AT, PS/2, 386/486, and compatible machines and the instruction sets for the 8250 and 16450 UARTS. Comm++ currently supports the Zortech C++ and Borland C++ compilers.

M&T Books

501 Galveston Dr.
Redwood City, CA 94063
(415) 366-3600

M&T Books' latest offering provides novice to intermediate C and C++ programmers with a complete guide and tutorial to Borland C++, the new object-oriented programming environment for developing DOS and Windows applications. Clean Coding in Borland C++ presents clear, easy-to-understand explanations of Borland C++ features and functions plus complete coverage of the powerful tools included with it.

Profile/V

First Class Software
P.O. Box 226 Boulder
Creek, CA 95006-0226
(408) 338-4649

First Class Software announced **Profile/V,** an efficient, interactive performance analysis tool for Digitalk's Smalltalk/V Mac and Smalltalk/V 286. Profile/V helps programmers get the most out of Smalltalk/V by showing where time is being spent, which methods are most expensive, and which statements within each method are costliest. Profile/V also includes a filtering system (called gathering) that helps users profile the recursive methods common in object-oriented programs. Profile/V comes with an extensive tutorial explaining optimization strategies in Smalltalk and showing users how to identify and exploit opportunities to make code faster.

Send Product Announcements to Marie A. Lenzi, 225 Rector Place, No. 12N, New York, NY 10280. Fax (212)786-4032. Include company name, address, and phone number.

Listing 1. "Hello, world!" for Windows.

```
program Hello;
uses WObjects, WinTypes, WinProcs;

type
        THelloApp = object(TApplication)
            procedure InitMainWindow; virtual;
        end;

THelloApp.InitMainWindow;
begin
        MainWindow := New(PWindow, Init(nil, 'Hello, Windows'));
end;

var HelloApp: THelloApp;

begin
        HelloApp.Init('HelloApp');
        HelloApp.Run;
        HelloApp.Done;
end.
```

database or virtual memory management system, persistent objects are handy for restoring the state of an application or for bringing data across applications. The stream types allow us to easily store objects, or collections of objects, out to disk. For example, to store the state of any application to disk, you simply store the desktop collection. All of the supplied objects can be written to disk by responding to a Store message and loaded from disk with a Load message. User-defined objects can implement their own load and store methods also by storing out each of the components in a byte stream.

TRADEOFFS

An application framework requires a kind of socialist perspective: one must balance the needs of the many applications against those of the few. Our designs focused on the needs of the typical PC programmer who develops applications for others. This ranges from the lone programmer developing in-house applications for his company to the professional developer of commercial "shrink-wrap" applications. Although it is tempting to add high-powered features that appeal only to the high-end developers, we chose to keep the application frameworks "lean and mean" while keeping them open ended enough for professionals. Both designs are efficient in both speed and space. They provide what every program will need, but have little "fat" that will prevent programmers from using them for applications that must be fast and memory efficient. The typical cost is about 15–30K for a full-blown Windows application.

ASSESSING APPLICATION FRAMEWORKS

The emphasis of an application framework is on creating applications. In turn, each application framework has built on past developments to provide increased functionality for application developers. The more standard the application, the more likely you will benefit from using an application framework. If you need to provide applications with a common user interface — windows, dialogs, event handling, and traditional data management using arrays, linked lists, and file management — an application framework can save you considerable time and effort. However, if you need to write low-level system software, one-shot utilities or other types of programs, an application framework probably won't be of much use.

By their design, application frameworks require the use of object-oriented programming techniques. For many programmers, this may be the first solid proof that you can "inherit" a fully functioning application and not give up the efficient performance of a truly compiled language. We have found through informal research that using application frameworks can reduce coding by between 25% and 50% and results in easier maintenance of programs.

Since application frameworks are by their nature open ended and extensible, there will no doubt be additional high-level libraries built on top of these frameworks. This will help us realize the full potential of object-oriented programming and hopefully save us all from endlessly reinventing the window, the application, and other generic components. For example, The Whitewater Group offers the ObjectGraphics library, which adds a rich set of platform-independent graphics objects to ObjectWindows. By building upon a base-level foundation of an application framework, it becomes much easier to combine class libraries from different vendors, something that has never been easy to do with traditional function libraries.

Application frameworks are by no means a panacea. However, they are the next logical step in achieving two important benefits of object-oriented programming: productivity and easier maintenance.

In an effort to promote standards for object-oriented development on graphical environments, ObjectWindows has been offered to the Object Management Group for adoption by other vendors. The specification is available either from the Object Management Group or from Borland International. ≡≡

Zack Urlocker is the Turbo Pascal product manager at Borland International and has taught object-oriented programming to programmers worldwide. He can be reached at Borland International, 10 Victor Square, Scotts Valley, CA 95067.

LANGUAGES

Report on the seventh meeting of the Object-Oriented COBOL Task Group

by Megan Adams & Dmitry Lenkov

This column reports on the progress at the June meeting of the Object-Oriented COBOL Task Group (OOCTG), which has been chartered to produce a standard specification for Object-Oriented COBOL.

Megan Adams has been a software engineer on the COBOL compiler project in the California Language Lab for seven years. She has been HP's primary member of the CODASYL COBOL Committee, the standards body that designs new features for COBOL, and an alternate on the ANSI X3J4 Committee, which adopts and publishes ANSI Standard COBOL. Adams also serves as Vice-chair of the Object-Oriented COBOL Task Group.

Dmitry Lenkov has over twenty years experience in development of compilers, tools, and program development environments, and ten years of experience as a language designer. He is the chairman of the ANSI C++ Standard committee and serves as a participant of the CODASYL Object-Oriented COBOL Task Group. He is currently employed by Hewlett-Packard.

It also introduces the concept of and rationale for Object-Oriented COBOL and reflects the understanding of Hewlett-Packard participants in the OOCTG. It is not intended as an official Task Group document.

In November of 1989, in response to an HP initiative and to the growing interest among COBOL vendors and the COBOL-user community, the CODASYL COBOL Committee sponsored a symposium on Object-Oriented COBOL. At this meeting, the Object-Oriented COBOL Task Group was formed with the charter of drafting a specification for Object-Oriented COBOL. The OOCTG has met quarterly since January 1990, and its participants include both vendors and users (such as IBM, HP, Hitachi, Realia, MicroFocus, Digital, Sun, Southwestern Bell, and Hewitt Associates).

ADVANTAGES OF OBJECT-ORIENTED COBOL

Before discussing specific features and concepts that are under consideration or have been included in Object-Oriented COBOL since the June meeting, we will explore the question, "What does this technology offer COBOL users in particular?"

The advantages of object orientation have been covered in a previous article.[1] Here, we discuss the advantages of using Object-Oriented COBOL as a vehicle for the migration to the new technology.

Object orientation is a technology that can be effective at addressing the central problem of the high costs of software development and maintenance. Object-Oriented COBOL provides the advantages of object orientation by building into the language constructs and concepts that support powerful recent ideas about increasing software productivity and quality and, thus, help to reduce costs. For example, code reuse can provide the most significant single improvement in software productivity. Object-Oriented COBOL supports code reuse with the mechanism of class inheritance. Also, Object-Oriented COBOL adds a sophisticated modeling capability based on objects, classes, and interfaces. This modeling capability makes it easier to handle increasingly complex application requirements. Maintenance costs are also addressed. Interfaces in Object-Oriented COBOL support modularity, which eases maintenance because changes in specification are limited by the interface to having only a localized effect. An important side effect of modularity is the improved quality of code, which in the long term can reduce maintenance costs. A significant portion of maintenance costs is derived from the need to modify existing programs to deal with changes in specifications or enhancements.

Interfaces and class inheritance in Object-Oriented COBOL provide a means of making it easier to extend existing functionality to handle changing requirements.

There are many advantages to providing object-oriented capabilities in COBOL. Providing these capabilities in COBOL rather than forcing COBOL users to migrate to a new language provides an evolutionary path to a powerful new technology. Also, adding these capabilities to COBOL via the COBOL standards process ensures that COBOL users will continue to have the benefits of portability and consistency across implementations that COBOL has previously maintained. Finally, it can be argued that object orientation is particularly suited to the problem domain characteristic of COBOL applications because

... the impending Big Blue-Big Red alliance may be less an anomaly that a natural and salutary culmination of three recent trends in the personal computer business: growing buyer rejection of single-source proprietary equipment; a strong movement toward industry-wide standards and "open systems" after a decade of rampaging technological progress and proliferating incompatibilities; and a growing recognition that the future of office computing will be shaped by software design, not name-brand hardware sales ....

IBM-Apple deal anticipates new PC era, Curt Suplee, ***TORONTO STAR,*** *7/14/91*

... But IBM and Apple are being drawn together by a common enemy — Microsoft Corp., which through is software programs and alliances is apparently trying to dominate every segment of the computer industry worth preserving or developing for the next century. Additional pressure to unite comes from the fact that the $35 billion a year PC industry is in the midst of an unprecedented price war/restructuring. Both IBM and Apple have been forced to repeatedly slash PC prices just to hold onto market share. And computer users — no longer the docile sheep they once were — are doing everything possible to promote "open systems" that avoid dependence on a single computermaker or software developer. Faced with a formidable enemy, weak markets and more demanding customers, neither IBM nor Apple can afford the cost of developing tomorrow's "hot box" technology all alone ....

Apple and IBM talking about technology swap that mirrors industry flux, ***EDP WEEKLY,*** *6/17/91*

... IBM and Apple face a hard slog both in trying to create such an environment and in getting others to accept it. Neither company has a strong track-record in systems-software development. Though nobody can be sure what secrets lurk in corporate research labs, IBM and Apple seem to be lagging behind firms like Hewlett Packard, Sun, Digital and a variety of others rushing to develop their own object-oriented network technologies. Though IBM and Apple say they will license to others any software they develop, they will keep firm control of the technology. Smaller companies may be chary of building into their products any dependence on the proprietary technologies of two large and sometimes paranoid rivals. Besides, though most successful partnerships are a marriage of equals, this is anything but ....

... access to IBM's vast resources of cash, technology and salesmen must seem like a godsend to Apple. The first problem with the deal, however, is that IBM has nothing like the same burning need to make the joint venture a success. Though the projects could be lucrative, so could any number of other pies in which IBM has its fingers. Indeed, IBM has several partnerships covering much the same technical ground as it intends to explore with Apple, including licensing deals with NeXT, Metaphor, Microsoft, Borland, and other computer firms. And the deal with Apple puts IBM in an ideal position to play off Microsoft and Apple, the two dominant suppliers of systems software for personal computers, one against the other.

IBM and Apple: hold hands and hope, ***THE ECONOMIST,*** *7/6/91*

... With DOS and Windows based on the foundations of outdated technology, OS/2 could represent a significant threat with the marketing clout of IBM combined with the technical expertise of Borland International. Don't be surprised if Borland eventually takes over control of OS/2 completely, with IBM endorsing it from the sidelines. IBM has never been known for writing decent programs, but Borland has already established itself as one of the best ....

The benefits of free enterprise, ***W.E.W., COMPUTER EDGE,*** *6/21/91*

... IBM's recent maneuvers in the desktop operating system market are taking on all the intrigue of ancient Rome. Former friends (Microsoft) are now enemies while former enemies (Apple) are fast becoming friends — more or less .... Still, other hard-charging rivals are being drawn into IBM's orbit ....

IBM reaches out to the industry, ***INFORMATION WEEK,*** *7/1/91*

... Borland President and CEO Phillippe Kahn hinted that his company probably will jettison most of Ashton-Tate's other [than dBase] products. Although he would not disclose details of his plans, Kahn stressed Borland intends to use its newfound strength to butt heads with relational database giant Oracle Corp .... Along with dBase, a likely survivor of a Borland selling spree would be Interbase Software Corp., recently acquired by Ashton-Tate .... Other developers said they will wait to see if Borland invests its technical expertise. "Product releases speak louder than company combinations," said Tom Shanahan, chief operating officer for SBT Corp ...

dBase, Interbase Server cream of A-T Crop, Karen D. Moser, ***PC WEEK,*** *7/15/91*

... In addition to doubling its revenue, the merger adds Ashton-Tate's huge base of dBase customers and fills holes in Borland's product line by giving Borland access to Interbase, a multiuser Unix database, and word-processing applications .... Borland's "Object dBase" is part of the coming wave of object-oriented applications and systems software. The development of these products has made bedfellows of ... unlikely partners ....

Borland buys Ashton-Tate, fills holes in database line, Margaret Ryan, ***ELECTRONIC ENGINEERING TIMES,*** *7/15/91*

... "They're going to create a market conflict between the two products," said Terry Quinn, an analyst with Kidder Peabody .... "You still have two products that are essentially incompatible .... While you can access data, you can't run a dBase development program and run it on Paradox. The marrying of the technology is at the file level."... It also upgrades Borland from a recent upstart in the software business to a legitimate competitor against market leaders Microsoft and Lotus .... Mr. Kahn appears eager to snap up additional market share as quickly as possible, and may have been worried that troubled Ashton-Tate would prove an attractive takeover candidate for another software house ....

Borland to Buy Ashton-Tate, Brian Deagon and Andrew Collier, ***ELECTRONIC NEWS,*** *7/15/91*

Industry News

... When asked whether Metaphor's object-oriented technology is intended to set the foundation for the proposed software firm, an IBM spokesman replied: "I wouldn't call it the foundation. It will be a valuable part." IBM, which took an 8.3 percent stake in Metaphor for $10 million two years ago, had been co-developing a line of object-oriented software as part of a joint venture with the Mountain View-based firm. The joint venture, Patriot Partners, which reportedly included 175 Metaphor employees, was developing software that would run on a number of undisclosed computers and operating systems. Sources close to the companies said the software under development at Patriot Partners would not have been available in the marketplace for several years, while Microsoft and Apple were believed to be much further along in their efforts to devise the object-oriented technology for a new generation of RISC and CICS platforms ....

IBM to Buy Metaphor; see Apple Tie, ***ELECTRONIC NEWS,*** *7/15/91*

The alliance of IBM and Apple Computer, Inc. is not only likely to escape an antitrust challenge, it could also keep the regulators away from archrival Microsoft Corp.'s door ... Norman Weizer, a computer industry analyst at ... Arthur D. Little, Inc., agreed. "This latest move probably eased the antitrust tension in the industry ... if there was any thought of an antitrust investigation of Microsoft brewing at [the US Department of] Justice, there won't be now." The IBM/Apple combination, he said, will pack more than enough of a competitive wallop to dispel any view of Microsoft as an unfairly heavy hitter. However, legal experts said nothing in the IBM/Apple scenario should affect the FTC ongoing investigations of Microsoft and chip master Intel Corp. "The FTC challenge to Microsoft isn't being made on broad policy grounds," said a former Justice Department antitrust litigator who asked to remain anonymous. "This is a very targeted investigation; the FTC received so many [separate] complaints about specific Microsoft practices that it had no choice but to look into them."... The attorney also warned against confusing the recent wave of FTC interest in policing high-technology business practices with a resurgence of anti-trust enforcement on the Justice Department's part.

IBM/Apple deal could take heat off Microsoft, Nell Margolis, ***COMPUTERWORLD,*** *7/15/91*

As talks between Apple and IBM continue, industry analysts believe that the two former adversaries have been brought together in part by the fear that consumer electronics giants such as Sony and Nintendo could infiltrate the PC market. Consumer electronics companies are widely expected to leverage their market share, potentially challenging PC manufacturers for the business computer market. Consumer electronics firms churn out products at a much faster rate than computer manufacturers, and their products are typically lower priced .... But though consumer electronics companies have designs on entering the business PC market, Apple has a few ideas about the consumer electronics market. It is reportedly teaming with Sony to develop a 1-megabyte CPU outfitted with a CD ROM drive that could be hooked up to a television set and could run Macintosh software.

Home electronics giants bring IBM, Apple together, Kristi Coale, ***INFOWORLD,*** *7/1/91*

...Last week [IBM] revealed a staggering succession of partnerships with Borland Int'l Inc., Lotus Development Corp., Candle Corp., Goal Systems Int'l, Inc., Bachman Information Systems, and Platinum Technology, Inc. A pact with Apple Computer, Inc. is tantalizingly close. Two weeks ago, Wang Laboratories, Inc. agreed to resell IBM hardware. Turn the page back to February and there is IBM and networking leader Novell, Inc. shaking hands...Big Blue has added layers of insulation between itself and Bill Gates. These pacts don't just reduce IBM's dependence on a partner that has been accused of betraying it by pushing Windows instead of OS/2; they also directly challenge Microsoft Corp. in key markets. But IBM's deal-making also has larger motives. There are certain product areas IBM feels it needs help with: object-oriented, cooperative processing, networking, and systems management. As vital as they are, IBM's corporate culture could well destroy these fragile partnerships before they can flower. The Apple deal may yet flounder because of a deep-seated fear of Big Blue bureaucracy... IBM can't afford to let that happen...

... IBM, it has been said, maintains its power in the computer industry through the FUD factor — the ability to manipulate markets, customers, and competitors by creating a climate of fear, uncertainty, and doubt. The computer giant, the theory goes, is the only safe choice in such a confused environment .... Given IBM's size and the often competing goals of internal groups, customers have legitimate concerns about whether the various agreements share a common purpose .... A few years ago, IBM pledged support for open systems by joining the Open Software Foundation. IBM did a lot of posturing, but little has come out of the OSF. Now, the Apple venture raises questions of how the OSF fits into this new relationship. At PC Expo recently, the company did a remarkable job of building momentum for its version of OS/2. Now, only a few weeks later, that momentum has been slowed because customers are wondering when and around which standard IBM's desktop strategy will coalesce. Until IBM provides answers, customers should be wary of long-term commitments to IBM.

The FUD boomerang, Ira Sager, ***COMPUTER SYSTEMS NEWS,*** *7/22/91*

... In addition to Price Waterhouse, which recently deferred purchases of Microsoft Windows to further evaluate OS/2 2.0, IBM vice president and general manager for marketing and business development Joe Guglielmi said large-volume corporate customers had committed to or were evaluating OS/2 2.0. Those companies include Nippon Telephone & Telegraph, Frito Lay and Royal Bank of Canada ....

OS/2 2.0 dominates PC Expo as Microsoft skips show, Heather Clancy, ***COMPUTER RESELLER NEWS,*** *7/1/91*

this approach is data centered, and COBOL applications are primarily engaged in some form of data processing.

The advantages of taking an evolutionary path to Object-Oriented COBOL are many. For one, this approach minimizes the retraining of the existing large base of COBOL programmers. COBOL programmers will need training in the new constructs and concepts and also in a new design approach, but they can use their existing COBOL expertise as a base. As there are estimated to be three-million COBOL programmers in the world today,[2] this approach is clearly more realistic than retraining these programmers in a new language. (Also, it is doubtful that a language such as C++ or Eiffel would provide adequate support for some features that COBOL handles today, such as packed decimal support, file handling, and other constructs tailored to commercial requirements.)

Programmer training itself can be incremental, starting with an overview of the new paradigm, progressing to how to use object-oriented capabilities in normal COBOL programs, and finally to object-oriented application design, including the design of classes and interfaces. It won't be necessary to intensively train the whole staff in everything at once because the transition is naturally evolutionary.

Obviously, this approach also makes it possible to leverage existing investment in COBOL. Currently, over 70 billion lines of COBOL are being maintained and probably five to ten billion more are added every year.[3] It is unrealistic to require businesses to abandon this investment in order to reap the benefits of a new technology. A main attraction of this approach is that a relatively small amount of work is required in order to reuse existing code within applications using the extensions or to translate existing code into the extended language. The COBOL-user community needs to be able to choose from a set of options for migration to object orientation — the one that offers the best solution to their requirements. Object-Oriented COBOL provides the following options for this migration:

- Existing applications can define references to objects and use their new capabilities when they are enhanced during the maintenance cycle.
- Existing programs can be encapsulated as methods or as classes inside new object-oriented applications.
- New applications can be written and designed that are fully object oriented.
- New applications can be written and designed that include a mix of programs written in old-style COBOL and Object-Oriented COBOL modules.
- Eventually, Object-Oriented COBOL libraries should be available that provide significant software capabilities that users can access without having to code themselves.

Another advantage of this approach is that Object-Oriented COBOL is being designed in the COBOL standards community. The Task Group designing it is chartered to develop a standard specification. This means Object-Oriented COBOL will provide a standardized set of features across platforms and implementations. Also, Object-Oriented COBOL will be 100% compatible with existing COBOL.

> ***Programmer training itself can be incremental, starting with an overview of the new paradigm, progressing to how to use object-oriented capabilities in normal COBOL programs, and finally to object-oriented application design, including the design of classes and interfaces.***

A final advantage of this approach is that it incorporates new data-centered language constructs into COBOL whose problem domain is possibly more data centered than that of any other language.

At the most basic level, COBOL programs exist to process data and not to embody logic. Most COBOL applications are involved in some form of data processing. Their natural emphasis is on organizing and handling the data rather than functional decomposition based on procedural logic. Object orientation provides a data-centered approach as it supports defining objects that consist of data and of operations on that data. In object-oriented design, the structure of an application is suggested by the structure of the data. This makes object orientation particularly well suited to data processing applications and COBOL.

OVERVIEW OF INCLUDED FEATURES

Some of the most important basic features that have been included by the OOCTG in its draft documents are instances, classes, class inheritance, interfaces,

interface conformance, and factories. An overview of the first four of these features is included in an earlier article.[1] Factories and interface conformance are included in the next section.

OVERVIEW OF RECENT WORK

At the last two meetings, the OOCTG separated conformance of instance interfaces from class inheritance. Conformance provides the basis for generic code because it specifies when an object with one interface can be used where an object with another interface is expected. Saying that all office forms support a print interface is equivalent to saying that all office form instances conform to a print interface. Thus, they can all execute print functionality with a generic invocation, although the print may be implemented differently for distinct types of office forms.

> ***... basing conformance on interfaces allows generic code to be defined for entirely unrelated classes, which is not possible when conformance is based on class inheritance.***

Interface-based conformance is considerably more flexible than conformance based on class inheritance. In languages that base interface conformance on inheritance, derived classes are limited in their ability to redefine, delete, or otherwise alter the inherited services because these changes may break the conformance relationship. In Object-Oriented COBOL these restrictions are unnecessary and thus inheritance is a more effective tool for code reuse. It is easy to allow inheritance to define a conformance relationship, but it is not required. Also, basing conformance on interfaces allows generic code to be defined for entirely unrelated classes, which is not possible when conformance is based on class inheritance.

There was considerable discussion at the last meeting on the topics of factories. As a result, the Object-Oriented COBOL Task Group has introduced the concept of factory where the factory part of a class definition defines how new instances of the class are created. Factories can also define any data shared by all instances of a class. Compiling a class definition produces a factory, an interface to the class, and a set of common services for instances of the class. At runtime, new instances are created by requesting that the factory create them.

In other important areas, the OOCTG adopted syntax and semantics for the comparison and assignment of objects and object references. Also, the OOCTG defined a BOUND TYPE object reference that provides for dynamically allocated objects whose invocations are statically bound. Previously, only preallocated objects could have this most efficient binding.

SOME ISSUES OUTSTANDING

There was considerable discussion of issues related to the object lifecycle. Decisions on these issues have not yet been made but we expect they will have to be resolved fairly soon, perhaps at the September meeting.

The object lifecycle is a term that refers to the creation, initialization, finalization, and destruction of instances. Currently predefined methods (called, appropriately, CREATE, INITIALIZE, FINALIZE, and DESTROY) are specified in OOCTG documents to handle this functionality. There are some areas where the protocols between these methods need to be carefully defined so that anomalous situations do not occur. For example, if instances can be created and not initialized, or finalized but not destroyed, the instance has no consistent type.

A tightly defined protocol would prevent these situations from occurring, thus protecting users from a class of difficult bugs. For example, if INITIALIZE were defined to call CREATE, and DESTROY were defined to call FINALIZE, so that neither CREATE nor FINALIZE were visible to users, this class of errors would never occur.

Also, there are issues relating to how to define some of these lifecycle methods. If, for example, INITIALIZE were in the instance interface, it could cause conformance problems with other interfaces. However, if INITIALIZE were included in the factory and not in the instance interface, then interfaces could differ in their INITIALIZE prototypes and still be conforming. This seems desirable from a user point of view. ≡≡

References

1. Adams, M. and D. Lenkov. Report on the Object-Oriented COBOL Task Group, *Hotline on Object-Oriented Technology*, 2 (8), 1991.

2. Yourdon, E., Ed. Report on OOPSLA'89, American programmer, *Ed Yourdon's Software Journal*, 2(11), 1989.

3. Adams, M. and D. Lenkov. Object-oriented COBOL: powerful new capabilities in a compatible form, *Object Magazine*, 2(1), 1992.

... Not until the longer term, the latter-half of the 1990's, will evolutionary changes in the software process give way to truly revolutionary changes. Object-oriented techniques will influence changes in programming languages and methodologies. For example, object-oriented extensions to COBOL will be available in PC-based tools in 1991. The traditional waterfall life cycle model will be replaced by a software process model that is more formal (i.e., based on mathematical techniques), more amenable to automation, and emphasizes enterprise modeling, iterative prototyping techniques, system specification activities, and requirements traceability. Artificial intelligence techniques such as expert systems, inference-based application development methods, and natural language processing will be embedded into CASE tools. Structured methodologies will be extended to include object-oriented methods. Software reusability will replace development strategies that build software systems from scratch ....

First evolution, then revolutions, Carma McClure, ***CASE TRENDS,*** *Special/Summer 91*

... it's a little like the optimistic small boy who woke up on Christmas morning to a pile of manure: there just has to be a pony in there somewhere. In fact, proponents of object-oriented programming think they hear a whole stable...And OOP should not be confused with visual programming, although they are related .... Borland International recently launched a product called ObjectVision, designed to let end-users create "programs" without writing program code. John Mandell, product marketing manager for ObjectVision, says it is a visual programming product, not an object-oriented language, but "we use the object model to get to the visual programming." The real point of ObjectVision, Mandell says, is to let end-users do things without learning to program. But that is not a benefit of true object-oriented programming. Assuming that one can dig through all the hype and get to the horse, the next question is how to ride it. The key point seems to be that some learning — and perhaps some unlearning — is required .... In other works, using an object-oriented language isn't likely to give you the promised benefits unless you learn to think in the appropriate way to take advantage of it.

OOP is more than just a buzzword, Grant Buckler, ***COMPUTING CANADA,*** *6/20/91*

HP recently decided to withhold release of a Unix version of its NewWave Object-oriented operating environment until the Object Management Group rules on the HP/Sun proposal for object management .... By waiting until after August, when the OMG will rule on the HP/Sun proposal for their Distributed Object Management Facility, HP will be able to incorporate new standards into its subsequent releases of the software developer's kit and NewWave for Unix .... HP also is prepared to deal with OMG's standards if they derive from other vendors....

HP waits for OMG ruling before releasing NewWave kit for Unix, Brian Simpson, ***WORKSTATION,*** *7/91*

... Only the interpreted object-oriented systems such as Smalltalk, object-oriented Lisp, and various proprietary object-oriented development systems, have a clear edge over Nextstep in programmer productivity. Because these systems are unsuitable for producing commercial applications due to their poor performance, huge size, or restrictive licensing policies, it is hard to refute the commonly heard claim that the Nextstep environment is the most productive mainstream development environment available today....

The next NeXT, Scott Raney, ***UNIX WORLD,*** *7/91*

... Smalltalk is not about to replace COBOL, but it is finally maturing into a viable choice in application development, especially for users looking for a tool to speed development of advanced graphical user interfaces in client/server applications...As a dynamically compiled language built on reusable objects and a virtual interface that uses machine-independent, intermediate code, Smalltalk is also easily portable between the platforms it supports ....

... there is still no widely accepted development methodology for Smalltalk or for any other object-oriented environment. In addition, many users are still making the transition to the relational model and structured programming techniques. "Most [IS developers] still don't know what to do with objects. They're still traumatized from making the migration to the RDBMS," says Natasha Krol, application program director at the Meta Group in Stamford, Conn. Smalltalk also faces increasingly stiff competition not only from other object-oriented languages such as C++ bur also from new GUI-building tools, such as Easel from Easel Corp, and Actor from Whitewater Group ....

Smalltalk grows up, Jeff Moad, ***DATAMATION,*** *7/15/91*

... The most sought-after goal for the future will be establishing a standard class library, and the ideal language will include that library, according to Scott Moody, an industry consultant .... Application developers could then avoid the problem of being tied to one vendor's class library, although problems with incompatibilities between languages would remain. Languages that lack a foundation class library or an application framework open the door to the incompatibility problems commonly encountered today .... Industry analysts expect sales of class libraries for plug-and-go applications to flourish, providing plenty of incentive for vendors to resolve class library incompatibility problems. When class library compatibility is no longer an issue, aerospace suppliers and other companies that undertake large-scale projects involving numerous different components should prove good generic class library customers...

Object Orientation's uphill struggle continues, Sarah E. Varney, ***DIGITAL REVIEW,*** *6/17/91*

... But being tied to the older technology of MS-DOS could also be a drawback for Windows. So Microsoft is developing a new operating system, known as Windows NT ... NT — which stands for new technology — will run on powerful personal computers using different kinds of microprocessors and will be better geared for use in networks. Indeed, Microsoft plans a whole series of version of Windows for different machines, ranging from pocket-sized personal information organizers to supercomputers. The company is also looking at using Windows to help control computerized facsimile machines, voice-mail systems and high-definition television sets ....

One day, junior got too big, Andrew Pollack, ***The New York Times,*** *8/4/911*

Analysis & design

Have you read the new book by Pope John Paul, "Why I became a Protestant"? Probably not, but if you want to read a book about a remarkable metamorphosis, try "Object-oriented Analysis" by Peter Coad and Edward Yourdon. In it Yourdon renounces "Yourdon Structured Analysis" in favor of Object-oriented Analysis (OOA). The "Yourdon" method, based on data flow diagrams, is the most widely-used approach to system analysis in North America. "Neither author has drawn a single data flow diagram in the past few years, except (on rare occasion)" they write... "For us," the authors conclude, "OOA is the future, and the future is here and now .... We see the 1990s as a period of gradual acceptance of OOA."

Authors renounce structured analysis in favor of OOA, John Lejderman, ***COMPUTING CANADA,*** *6/20/91*

... Myth #4: Object-oriented techniques make CASE obsolete. This is nonsense. If anything, the advent of object-oriented programming languages that incorporate control structures, data structures, and inheritance hierarchies make language-support tools essential. In the future, analysis and design will become dominant activities....

... As we become smarter about reuse technologies, fewer of us will do "traditional" programming. Most of our work will be to understand system requirements and architectures to effectively use existing components. Standardized methods supported by rich design environments will be critical to managing the complexity of these systems and communicating and preserving our design intent ....

Don't fall for these six CASE myths, Lou Mazzucchelli, ***EDN NEWS,*** *6/13/91*

The modularity of object-oriented programs is not only beneficial down the line, it makes programs far more testable. Object methods seem to be shorter and more focused than functions produced with a nonobject mindset. Object-oriented programs seem to have fewer global variables and more intuitive designs. So far, not many people have inherited object-oriented code to maintain, but the few I've talked to bear out this view ....

Up the revolution, Larry O'Brien, ***COMPUTER LANGUAGE,*** *7/91*

... If you have been given an imposed deadline, try a "timebox" approach. Work backward from the imposed date. Divide the work into "major chunks." Assign each "chunk" a specific amount of time. Make it quite clear in your plan and assumptions that when a chunk's date has been reached the team will go on to the next chunk, even if the work is not completed for the previous chunk. This approach is useful when you are "piloting" a new technology. It recognizes that the first time through you are learning the basics and will not be proficient in all aspects of the new technology. It help to avoid getting "bogged down" in one aspect of the technology ...

Point/Counterpoint: project estimating considerations for object technology — The manager's perspective, Rosemary Kurtti, ***OBJECT MAGAZINE,*** *7-8/91*

GUIs

... Object-oriented programming is probably the best way to build and maintain GUI applications .... It may take you up to six months to become proficient with an object-based environment and the vendor's class library. Much of this time will be invested in trying to understand what may appear to be a frighteningly large collection of vendor-supplied classes. Most beginners feel overwhelmed by hundreds of new classes, each with its own name, purpose, and messages. However, the consolation is that you can start out with a few classes and build your knowledge base gradually ....

Get GUI, Robert Orfali and Dan Harkey, ***COMPUTER LANGUAGE,*** *7/91*

Languages & tools

... Smalltalk/v has been realized in DOS, Macintosh, and Windows versions. Much of the code can be used across all the environments. Objects can be stored in text form, and "filed out" and "filed in" from system to system. Because Smalltalk is an interpreter, these objects can be introduced into a running system. The potential exists to network together Smalltalk systems on a number of different platforms, and to let them exchange objects in real time. This is not something you can do with C++. Smalltalk/v has the most thorough tutorial of any of the packages reviewed here. The manual is a complete course in the language, and the example files give you working code for most applications. Smalltalk is not like other computer languages. Instead of being like a musical score, it is more like a jam session in which you create both new instruments and new musicians as you go along. When you've constructed your band, you're ready to play. Smalltalk is extraordinarily interactive, and the ideal environment for creative people. Accessibility is excellent, limited only by Smalltalk's unusual syntax. Every element of Smalltalk, from object creation to debugging and running the application, takes place with a single Windows application. Integration is total. The assistance to clear thinking, Smalltalk's clean handling of the Windows environment, its integration and rich data taxonomy, and its potential for interplatform development, make Smalltalk/v the winner among all the packages we have surveyed....

Breaking into Windows, Birrell Walsh, ***MICROTIMES,*** *6/19/91*

... Perhaps the best way to understand event-driven programming is by comparison with the more traditional approach. In a conventional program, while waiting for user input, the entire application waits, looking at each keystroke or mouse action to figure out what the user intends. In an event-driven application, objects are defined to respond to a single stimulus (keystroke or mouse action). A run-time event manager (a sort of command supervisor) evaluates user input and decides which objects to activate. The end result is that each object neither knows nor cares if the user selected a different window and didn't come back for hours. A program like Windows (or any application that supports simultaneous activities), requires an event-driven approach ...

Document architecture standards of the '90s, Mike Cunningham, ***LIBRARY SOFTWARE REVIEW,*** *6/9*

GETTING STARTED

Getting started with object technology: effectively planning for change

by Mark Lorenz

The advent of a new technology is always accompanied by the trails and tribulations of coping with the necessary changes in the way we do business, both technically and managerially.

For the last few years, I have been working on team efforts to develop products using object technology. As we broke new ground, we learned some lessons that have been helpful for those that have come after us.

During these efforts, the team members transitioned through various stages of expertise. Before getting into the start-up steps, it is worthwhile to prepare you for the metamorphosis your people will undergo. This will shed some light on the advice that follows.

STAGES OF OBJECT EXPERTISE

Novice

At this early point, we try to get our function-oriented heads turned around so that we can see the application objects. We have a hard time finding the objects, even though we look hard and long. This stage results in a lot of functional code disguised as objects. It's best to have a Guru along at this stage, but more about that later. The Novice stage typically lasts about three to six months.

Apprentice

At this stage, we start to understand what it means to be object-oriented. We hear a lot of "ah-ha!"s and "I see!"s. We may still be more comfortable working in small teams since the interaction has a creative dynamic all its own. We gain more confidence. We do some good (and not so good) designs during this time and are still fairly dependent on the Guru. This period lasts another three to six months.

Journey Man

We now start to work on our own. We have internalized the object paradigm, but we still rely on a Guru during mental roadblocks and design reviews. The duration of this period varies greatly depending on the person, but probably lasts a year or more. Seasoned Journey Men work well and independently and are distinguished from Gurus only by the speed and inventiveness of their work.

Guru

This is the stage where objects come naturally. We automatically see things in an object perspective and, in fact, may have trouble seeing them in any other way! We move quickly through the phases of development, exploring during rapid iterations. Somehow, we act as "enzymes," causing project "chemical reactions" to occur — moving the project over hurdles toward completion. This stage lasts the rest of your life, if you're good enough and lucky enough to make it.

I know this stage exists because I have seen what are easily recognizable Gurus. I aspire to someday consider myself an object Guru — according to one's own definition. To me, there are only a handful of Gurus to be found. I would argue that they are blessed with the right combination of intelligence, personality, perspective, and experience to be the "fittest" object animal in the software jungle. The rest of us may have to rely on the evolution of proven processes and methodologies to perform effectively and survive our project development efforts.

These stages became pretty clear for us over the years. You can tailor the advice that follows according to the category of people you have on your staff.

One thing that's for sure — once you go down an object path, you won't want to go back. I've yet to see someone successfully make it past the Novice stage and even consider thinking about systems in other than object terms.

Mark Lorenz has been developing software systems for over fourteen years. He has worked on a variety of platforms, languages, and applications. His efforts since 1987 have been largely devoted to the area of object technologies, tools, and techniques. He works for IBM in Cary, NC.

Suggested steps for a good start

The following steps are suggested as tips on how to get a sound start at effectively utilizing object technology. These steps were documented during a look back at where we came from, where we ended up, and how we got there.

I have presented this information to a number of companies taking their first steps down the object path, including large companies from England and the U.S. These points have been favorably received, but it remains to be seen how effective the lessons we learned apply to the efforts of others.

Get training

This may seem self-evident, but it is important enough to articulate. Object development is not the same as function-oriented development. It is not the same as your FORTRAN programmers learning to program in C (although some may consider that as harrowing an adventure!).

You should plan on having two to three weeks of informal training, followed by a week of formal training. The overall learning curve is around six months. Remember, management needs training too!

Get resident object expertise

It is essential that you have object expertise on site. If you don't have in-house experts, hire some. It is essential for good designs and code (i.e., success).

There are good consultants out there that can make sure you get off on the right foot. Be careful though, "object" is the buzzword of the day, and not everyone who claims knowledge is really knowledgeable. A quick escape clause for a first contract go-around is a good idea.

Once you have grown your own object experts, you can wean yourself off the outside consultants. At this point, you should establish a mentor program with the most experienced object people playing "big brother" to the object Novices. By the way, a twenty-year software veteran with no object experience is an object Novice.

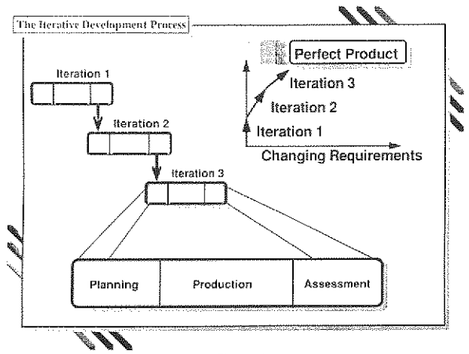


Figure 1.

Use an object methodology

There are now a variety of books published on object analysis and design (see the references at the end). The point is that you should have some methodology to follow. People work better and you have a better chance of being successful (i.e., delivering the right product at a competitive price) if there are precise steps to follow.

Use an iterative development process

Traditional function-oriented development uses the waterfall approach. This approach has not led to a high percentage of successful projects since it does not plan for the ever-present changes in the marketplace and understanding of requirements.

To avoid these pitfalls, the preferred process is an iterative one.* The basic difference is that you set up a plan that includes major development iterations, with the next iteration's planning period adjusting the plan using the previous iteration's assessment period inputs, which is based on the results of the previous iteration's production period.

As Figure 1 illustrates, the iterative process is a controlled process and remains requirements driven. The ever-elusive "perfect" product is chased, as understanding of the problem domain, marketplace, and customer needs evolves. This process plans for change in the development plan iterations while moving toward a match with the requirements.

Use tools

There are not a lot of object development tools available today. Since object technology is relatively new to the mainstream commercial software world, it has not had the time for CASE tools and library systems to evolve.

Take the case (a-hem!) of CASE tools, with support for techniques such as DFDs, ERDs, and structure charts. The structured methodologies have been around for over fifteen years. Tools to support them have appeared only in the last six years or so, without much penetration in the software market.

There are object methodology tools beginning to appear, such as OOATool,† and group development tools, such as Envy,‡ but the market is still wide open.

You will need group development support tools first. You will also need tools to support your iterative

* For a complete discussion of the iterative process and how it relates to the software development phases and methodology, see *A Process and Methodology for Developing Object-Oriented Software Systems: A Practical Guide* by Mark Lorenz.

† OOATool, by Object International, Inc., is a graphical user interface (GUI) object analysis/design tool that supports the notation used in Peter Coad and Ed Yourdon's book *Object-Oriented Analysis*.

‡ Envy, by Object Technology International, is a multideveloper support tool for Smalltalk/V 286TM, V PM, V Windows, and V Mac systems that includes LAN-based versioning and class ownership support.

FYI

Excerpts from leading industry publications on aspects of object technology

Networks

As object-oriented software becomes more featureful and widespread, networking-industry observers say user-created compound documents and other objects destined for network transmission may come to contain information about their own bandwidth requirements, along with the usual facts about originating application and user. Within a few years, they say, such "bandwidth-smart objects" could help bandwidth-management systems on customer premises and in public networks allocate precious network capacity more flexibly and economically ...

Smarter objects?, ***COMMUNICATIONS WEEK,*** *6/24/91*

Databases

... The excellent performance of object DBMS products may be surprising to those veterans who have weathered the previous generations of database technology. When relational DBMSs were introduced in the 1980s, they were practically unusable for production applications because they were so much slower that the installed network and hierarchic systems. Initially, the relational products were applied in areas where performance didn't matter very much (e.g., decision support systems and prototype applications) but where their flexibility and relative ease of use were important. Over the years, the performance of the relational DBMS products has improved dramatically. They are now used in a wide variety of commercial applications. Even so, their performance has remained unacceptable for other families of applications, including many applications with complex data relationships as well astringent performance requirements. Even though the object DBMS products are relatively young, they deliver excellent performance — good enough for the demanding application areas of CASE, CAD, GIS, network management, concurrent engineering, and so forth ....

Integrating Objects with relational technology, Mary E. S. Loomis, ***OBJECT MAGAZINE,*** *7-8/91*

... at the International DB2 User Group conference ... Chris Date, of Codd and Date International, and Michael Stonebraker, a University of California at Berkeley computer science professor, separately told audiences that object-oriented extensions could be added to currently available relational DBMSs. "There is no reason why today's relational products can't be improved," said Date ....

Database gurus: no basis for RDBMS, OODBMS dispute, ***SOFTWARE MAGAZINE,*** *6/91*

... "Right now, OODBMSes are still selling to tire-kickers," said John Rymer, an analyst with Patricia Seybold's Office Computing Group, Boston. "People are still trying to understand [object orientation], which hasn't been helped much by the vendors, who don't explain it well and do a terrible job of distinguishing themselves from each other. But by late 1992, a much broader base of users will be involved. Objects support the databases people want to build."...

... Comparing five OODBMSes with two RDBMSes, [Rick] Cattell [an engineer with Sun Microsystems] found that the OODBMSes ran up to 40 times faster. The tests were conducted on applications that use a fixed working set of data, fit in main memory and involve objects that are linked by pointers...Cattell said that the purpose of the tests was not to show that OODBMSes are superior to RDBMSes overall but "to counter the marketing hype of the RDBMS vendors who say their products are superior to everything else. OODBMSes are superior for use in scientific and engineering applications, and that's what these tests measure." ...

OODBMSes emerge, Dan Richman, ***UNIX TODAY,*** *6/24/91*

Applications

... Furthermore, said Goal Systems' [Neal] Ater [vice president, research and development], "True integration depends on moving from an application orientation to a task orientation. If you think of system management in terms of objects, what becomes important is what task you perform, not what product you're in. This approach is our great hope for wiping out fixed product boundaries."...

3-D management framework: SystemView's shifting views, John Tibbetts and Barbara Bernstein, ***SOFTWARE MAGAZINE,*** *6/91*

Antihype

"If you start object-oriented programming today, don't expect to see the benefits of class reuse before two years go by." [Adele] Goldberg [CEO of ParcPlace Systems] says that it will take eight weeks of OOP "immersion" for experienced COBOL programmers to become productive and six months for them to become comfortable and expert. OOP is not magic, Goldberg said ... She bemoaned the fact that object-oriented technology is being advocated by "religious" devotees who have gotten "a little sloppy." OOP shortens coding time but places more emphasis on the analysis and design portions of the development cycle. Still, she conceded that "very few languages make it if there isn't a religious cult behind them."

Nanobytes, ***BYTE,*** *7/91*

PRODUCT REVIEW ≡≡

Application Organizer Plus Version 1.0

Instantiations Inc.
921 SW Washington, Suite 312, Portland, OR 97205

reviewed by David Bellin

Application Organizer Plus is a set of tools for developers working in the Objectworks/Smalltalk environment sold by ParcPlace Systems. It's a seamless product that fits right in with the Objectworks environment.

It is by no means a simple task to become a proficient Smalltalk programmer, although a rich class library and a useful set of provided browsers does help. The drawback to such an environment is that often they lack tools that support the work of programming teams on large projects. This is where AppOrg steps in.

> ***...this product is not for the casual user. For the professional Smalltalk community, it is almost a necessity.***

"Versioning" is provided for every method in the system. This consists of the author name, date, and time, or any other string defined by the user. In fact, a complete Version Browser is included that allows on-screen comparison of different versions of a method. In an example of the thoughtful and integrated design of AppOrg, version information can also be obtained from within other browsers as well, including the method lister, the inheritance browser, and the "package" browser. This leads me to discuss another major facility provided by AppOrg.

Packages are objects that describe changes to the Smalltalk image. When they are used, the change sets provided with Objectworks/Smalltalk are replaced. Packages define which classes or methods were modified or defined in a set; however, they do not contain the actual code, which is always in the image. The use here is in the management of complex projects. Packages can be used for Smalltalk project management since they can offer several independent sets of change records that can be viewed through the package browser. This is a start at overcoming one of the criticisms leveled at object languages.

The package organizer is a tool for grouping the changes that are included in packages and for distributing them to various packages. I find this feature to be of dubious value and prone to error. It is very easy to put unintended changes into the wrong packages. The Application Organizer manual itself admits this shortcoming, and it is hard to understand why then the package organizer was even included.

An Image Generator is included that assists in application generation, basically by means of allowing the programmer to define a set of classes and methods that are to be removed from the image file. Obviously, one must be careful not to take crucial classes out of the system (e.g., integers), but, barring such blatant bungling, this part of the Instantiations software works well.

At $495, Application Organizer is available for both IBM and MAC platforms. The documentation from Instantiations is well written, but this product is not for the casual user. For the professional Smalltalk community, it is almost a necessity.≡≡

David Bellin is with Pratt Institute, New York. dbellin @pratt.obitnet.

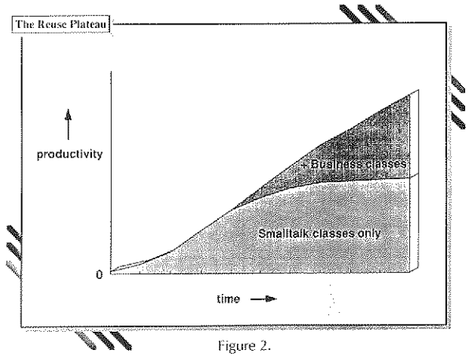


Figure 2.

process and object methodology. If you can, buy your tools. If not, make your own. Note that many activities can be supported without fancy tools. For example, collecting object information in the form of CRC cards* or in a flat file can take you a long way toward a good design. Keep in mind, a tool is only a facilitator and allows bad designs to be created as quickly and easily as good ones.

Make organizational changes

Besides your Development organization, the support organizations must change too.† This includes Test, Human Factors, and Publications groups. These groups must iterate along with Development.

Involve customers

It is a key benefit of object development that the iterative development and real-world modeling of applications allow customers to understand and be involved with what is being built. Involve them! After all, it is their system.

Create a reuse library

This is the key software asset of the company. It is absolutely critical to remain competitive in the software marketplace of the 1990s and beyond. Think of it this way — if you must create each new product from scratch at a unit cost of $200, and your competition reuses solid off-the-shelf components for up to 80% of the functionality in a similar product for a unit cost of $100, you are going out of business.

If you start out using a framework of reusable components, but do not add your industry-specific objects to the library, you will reach a productivity plateau (Fig. 2). This is because you are continually reusing the same abstractions rather than creating new abstractions specific to your business. It is important that you continue to reap the gains of reuse by extending the building blocks available to your application programmers.

A code reuse group, separate and independent from all project development organizations, needs to be formed to allow code to be made available to the entire organization. This "class clearinghouse" group would be responsible for facilitating a high level of reuse and minimizing reimplementation of function.

One of the possible techniques used by this group could be regular reviews of development materials, searching for abstract and nonabstract classes to include in an object library. It is even possible that this group could become more adept at recognizing and creating additional abstract classes since they see more diverse code and are exposed to multiple uses of the same types of functions. You may even want these people involved for a time on a project to facilitate the reuse of classes.

There are some key aspects to make the reuse library work:

- Project scheduling— The schedule must include time for the developer to create reusable components and to search for components to reuse.
- Incentives—The producer and the reuser of components should be rewarded. They are contributing to the quality and reducing the costs and schedule time of the company's products.
- Standards — See below.

Set up standards

Decide what informational content you want in your design documentation. This is more critical than traditionally.

Most of your time will be spent reading code, so you want to make sure that your coding standards keep this in mind. An industry expert (a Guru!) once told me he spends up to 75% of his time reading code! Why would this be? Because object development centers on reuse. Smalltalk development, for example, is essentially a series of searches for reusable components to be composed into interesting new combinations that are themselves reusable. However, these need to be certified through testing and standards to ensure they are solid building blocks.

Good object methods should be short! The average method size of our group's initial product effort (hundreds of classes and thousands of methods) averaged six lines of Smalltalk, and I think if we had done better object designs, the average would have been less! Long methods are a warning flag of function-oriented code or poor choices of objects and responsibilities. Also

* CRC stands for class, responsibilities, and collaborators and uses index cards to collect this information during analysis. The technique was originally proposed by Ward Cunningham and Kent Beck at Tektronix.

† For an extensive discussion of the organizational impact of object technology, see "Object-oriented development: not business as usual," by Mark Lorenz in *Hotline on Object-Oriented Technology*, 1(7), 1 – 4, 1990 .

remember, your screen real estate is limited so you want to make sure that the method code is immediately viewable in the source browser pane.

Use small teams

Due to the speed and volume of changes, larger teams will have communications problems. Break large projects into a number of small encapsulated teams. If you use subsystems and contracts between subsystems,* you can control the architecture (i.e., contract interfaces) between the teams, freeing them to rapidly iterate on their piece of the overall system relatively independently. This, of course, implies analysis legwork to ensure a fairly solid overall understanding of the architecture. This can be done via an analysis prototyping effort geared toward solidifying the requirements before the code development begins.

Use class ownership

Class ownership is key to change management, the build process, and code integrity. Changes submitted for a build can be on a class or application basis.

The class basis means that the class owner reviews changes if necessary and submits changes to his classes. This relates to code integrity. Historically, programmers making changes to the system "tromp" through whatever code necessary, making the most expedient changes to solve the "problem du jour" without focussing on the design of the "code du jour." With class ownership, the owner knows the design of the class, takes pride in its quality, and knows that he or she is on the hook for any and all problems.

An application basis means that a programmer can submit changes for all the classes related to his application within certain restrictions. He cannot, for example, delete any methods or instance variables, but is free to add them. This allows the programmers to focus on the application development at hand. Cyclically (about every third week), the emphasis shifts to a class basis for a week of "clean up." This is where the class owner focuses on his classes, making sure the designs are intact and robust.

Use a rapid build cycle

The iterations discussed earlier are separate and distinct from the build iterations, which are much shorter and more numerous. A typical build cycle has two software builds per week (versus an iteration, which may run for three and a half months). The speed of development snowballs as you go along. Longer periods for software submissions cause major headaches for the team. You need to establish procedures to minimize the build problems.

Architect for Change

By identifying the right objects in the application domain and giving them the correct responsibilities, you naturally end up with short methods and objects that are more reusable and flexible.

Another rule that carries over from the model-view-controller architecture is that the user interface (UI) should be kept separate from the underlying model objects. The UI changes much more often than the model. Also, many different applications can be built from the same model objects by changing the UI classes with little or no change to the model.

A STARTUP CHECKLIST

1. Get commitment from management for a pilot project that is important to the company with clear deliverables defined. If the project is not important to the company, no one will care how great your results are.
2. Get education for a small (three to six people) team and consulting expertise. Unless you have real object expertise on your project, you will not succeed.
3. Buy tools, especially for group development and methodology support. Areas such as sharing reusable components are not easy, but are key to your competitiveness.
4. Set standards and hold design reviews. Make sure you focus on good object models of the application domain, short readable methods, and on-line design information.

SUMMARY

Object systems effectively plan for change. Change is a fact of life. Our teams, using object technology, have seen orders of magnitude of improvement in productivity when some basic guidelines were followed. I have passed on these guidelines so that you can stand on my shoulders and not my feet. It is going to require some organizational changes to adapt to the implications of object technology. In the short run, we have to make do without the level of tool support we see in the function-oriented arena. The productivity potential is there, waiting for us to achieve it. ≡≡

References

1. Coad, P. and E. Yourdon. *Object-Oriented Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1990.
2. Wirfs-Brock, R., et al. *Designing Object-Oriented Software*, Prentice Hall, Englewood Cliffs, NJ, 1990.

* For a discussion of subsystems and contracts, see Ref. 2.

channel, a transmutation, a degeneration takes place: the managers lose appreciation for the essentials of the technical work. Essentials become implementation details. Technicians working in a vacuum start to lose touch with business needs. Decisions with critical business ramifications become implementation necessities. With the deadline pressure rising, the manager becomes the not-helping voice of the organizational anxiety. Technicians, those working toward the deadline, become the obstacles to the deadline. This is the moment when not the program, the project, or the software, but people, managers, and programmers mutate themselves into vampires, werewolves, and djinn. The paradigm shift of object orientation, by advocating a simple language common to both business and technology, gives a glimmer of hope. Despite the anxiety of the deadline and the typical neurosis of organizations, we can remain members of the same team, the team of human beings, and solve a few problems.

I claim object analysis methodologies, object design tools, and, in particular, C++ programming language as conceived and presented today have almost nothing to do with the paradigm shift toward object orientation. They are mostly business as usual. They are similar to the elements of an old regime disguised as revolutionaries, Securatate forces championing democracy in today's Romania. Why do we need yet another analysis framework if the language of the programming is high level enough to express the concepts of the problem? Why do we need more expensive bubble drawers if the modules of our implementation have the same articulation with the demarcations of our design? Why this bombardment of obscure terminology, polymorphism, virtual functions, friends, dereferences, protected, public, and private modes — if our determination is the determination to communicate? This hype is nothing but an industry for mental pollution.

The pseudo-science of structured methodology was evidently a sound compensation for the false start we generously call the software industry. The crutches should be buried now so that technology archaeologists can publish condescending papers in the future and get academic tenures. Nothing is more deceiving as saying in the same breadth: we had a paradigm shift. Everything has changed. Everything stays the same. That is the lucrative business of snake oil: if you ask the werewolves they are going to say, "Of course, there is no silver bullet!" ≡≡

continued from page 15...

existing parts for reuse. While this sounds fairly simple to achieve, try managing reuse "commissions" without the appropriate (CASE?) tooling.

Object based/object oriented

Recognize that the study system is object based, not fully object oriented. The primary difference is that there is a significantly higher level of precision and rigor applied to object orientation. If this much productivity can be achieved with a loose implementation of objects plus the proper tooling, think of what can be achieved with a more robust implementation of the object paradigm supported by tools that focus directly on identifying, developing, and then finding parts for reuse.

CONCLUSION

This study strongly suggests that object technology, supported by appropriate development and maintenance tooling can improve the quality and productivity of software developers. Furthermore, the timely availability of software products that directly support the strategic direction of the organization can directly contribute to improved short- and long-term bottom lines.

The study team went into the project to determine the improvement of "development productivity and software maintainability" offered by CASE. They learned something completely different. Now we need better tools and techniques to study and measure the impact of reuse on development. This should lead to more and better advancements in development tools and techniques.

Epilogue

The frightening last line of the study states that "since nearly all of these projects have been built by novices with the CASE tool, this offers an interesting opportunity to study the potential for reducing software development costs by staffing teams with less skilled developers." Why would they not consider a mix of experienced, skilled developers supported by less experienced (not less skilled) novices? Without the experienced, skilled developers, who will teach the novice? Can we cost justify the novice reinventing the learning wheel? Remember, the operative part of CASE is computer aided. With CASE tools a less skilled developer will produce, at best, an organized mediocre product. It has been proven that if you strive for quality, you get both quality and productivity. If you strive for only productivity, you get neither. ≡≡

SILVER BULLETS AND WEREWOLVES

On objects and bullets

by Adnan Adam Onart

Let us be completely open about it: there is a software crisis and it is polluting not only the quality of the products but the quality of our professional lives as well. What is a software project? What is an average day in a software project?

Quite often a nightmare! This phenomenon, the crisis in software is not caused, in my judgement, by the complexity or by continuously changing nature of software, but by our inability to communicate. As to complexity and change, were we not coping well for some millennia with this sizable complexity and change called Universe?

It is not the essence of software but some accidental characteristics of our folklore that lead us to a radical communicative deficiency: a social culture that uses language as a tool for domination rather than a bridge for relations; an education system that trains people to deal with abstract algorithms, to play with machines, and does not pay any attention to professional character formation; and a work pathos that pushes the dominating linguistic culture to its utmost limits. Even the act of information giving in this culture is an extreme form of information hiding. The very act of communicating suffocates communication. Question asking needs the disguise of questioning. "How come you are saying this?" is the standard melody of the question, never, "Could you explain this to me?" The very act of asking the question is, therefore, a road block to the answer.

An explanation, on the other hand, stifles the urge to ask a question. Sarcasm, such as the tone of the voice or a body or facial gesture ornaments answers. The speaker looks impatient or upset while answering a question. The subliminal message is nothing but, "It is very dumb of you to ask this question, nevertheless let me have the benevolence of explaining this to you."

That is it: as human species we seem to have a gift for comprehending abstract complexities and domesticating subparticles and quanta — please no eyebrows — into docile sheep. On the other hand, because of our inability to understand simple and concrete utterances, and because of our inability to perform trivial acts of question asking and answering, we are transforming straightforward payroll programs into werewolves.*

Object orientation — not as a new technology, not as encapsulation, inheritance, and polymorphism — as a professional paradigm shift — a perspective change in the social culture, education, and work habits — is definitely a silver bullet, a glimmer of hope. This new paradigm deploys itself as a shift toward a commitment, a commitment to communicate. To me that is the mission of the syntactically simple, but semantically powerful languages of the object. The same terminology and viewpoint can be used to understand the problems that need to be solved, to describe the solution strategies, and even to implement the solution: an Esperanto-like system that can be understood by everybody — implementors, designers, analysts, and, yes, even managers. This is the reason why objects are generating so much attention and also some extensive turbulence. It is not that this is a little better than that. There are professional, even personal interests at stake: established reputations and careers.

Here is the fairy tale of Ali Baba and the Forty Thieves: managers need to stay technically involved, not so much to supervise the technical correctness of the work, but to keep the communication channel open so that solutions can be found that are acceptable technically and businesswise. Without this open

Adnan Adam Onart, a professional werewolf hunter and works as a director of software architecture for Timeplex, Inc. He holds a PhD from the University of Istanbul. As an Alexander von Humboldt Scholar, he has conducted postdoctoral research in Frankfurt am Main. After several years of teaching and research at Middle East Technical University, Ornart joined the software industry. He can be reached at onart@timeplex.com.

* I suppose werewolves, silver bullets, and Count Dracula are not registered trademarks. Refer nevertheless to the following: Brooks, F. No silver bullet: essence and accidents of software engineering, *IEEE Computer*, April 1987, and Cox, B.J. There is a silver bullet, *BYTE*, October 1990.

STATISTICS

Object statistics on the way

by Marie A. Lenzi

To support the adoption of object technology, many organizations need hard numbers. Numbers that can be translated into dollars and transferred to the bottom line.

Currently, statistics for or against the new paradigm are difficult to come by. Objects just haven't been around long enough. So to satisfy the need, there is much extrapolation of existing figures from related statistical studies and a good deal of instantaneous creation of numerical support for the new technology. I marvel at those who I have heard make up "numbers" on the fly. I marvel even further at those who believe these stats. Numbers don't lie? Numbers can say anything you want them to say?

I recently came across a statistical study targeted at validating computer-aided software engineering (CASE) tools. However, on first reading it was clear that the study's findings supported not the productivity of CASE but the productivity of reuse. In this study, the mechanism of code reuse outshone the tooling almost into irrelevance. Almost! The CASE tool was indeed the mechanism that supported reuse. To reuse software you have to know where reusable parts are, how to use the parts, and, most importantly, that reusable parts must be engineered. Reuse would probably have been cut significantly had it not been for adequate tooling. But the CASE technology was not the reason the goods got to market, only the vehicle that drove them there.

Marie A. Lenzi is cofounder of Syrinx Corp. (US), which specializes in O-O technology, development, training, and consulting. She has extensive experience in systems development and methodology and is the Editor of Hotline on Object-Oriented Technology *and* Object Magazine.

CASE?

There is no clear definition in the marketplace as to what constitutes a CASE tool. I have heard text editors referred to as CASE tools. Frankly, back in the 1970s when TSO replaced 80-column cards, code management was improved by several orders of magnitude. This is not to diminish the value of CASE. I have worked in the area of structured techniques since the early 1980s. It wasn't until appropriate tooling became available around 1983 that the implementation of these techniques on any scale became viable. The physical creation, validation, and maintenance of the engineering models for software development are tedious, time-consuming processes easily supported by automation. Turning these tasks over to the machine leaves the analyst, designer, and software engineer free to focus on and to study the actual problem/solution space represented by the models.

CASE is not the engineering concept. CASE is the tooling, the automated assist that supports the engineering process. CASE is another concept, like AI and object technology, that has been market-hyped almost to its detriment. The CASE product manufacturers have marketed the tools as yet another silver bullet; the concepts supported by the tools were almost ignored. Initially, there was huge CASE product sales. Unknowing corporate customers bought into the hyped panacea of a tool only to find that a massive investment in additional conceptual education and alterations to the infrastructure were required. When the marketeers were found out, sales slowed considerably.

Don't get me wrong. I am a strong supporter of CASE, however, not as CASE for the sake of CASE. What I support is (1) a precise common language for communication between and among development teams, team members, managers, and, also, the end users of the products and (2) an organized repeatable set of steps that aid in the development of a clear problem/viable solution set. CASE does not do this. The concepts automated by the various CASE tools do.

SYNOPSIS OF THE STUDY

The study was conducted by Rajiv D. Banker, Carlson School of Management, University of Minnesota and Robert Kauffman, Stern School of Business, New York University and partially funded by the National Science Foundation. "The study is a report on an analysis of data collected in a year long field study of software development using integrated CASE at the First Boston Corporation, a large investment bank in New York City."

Recognizing the deficiencies in the current techniques of software development measurement, the study chose to use function points analysis, a technique that "gauges the size of an application in terms of the [system's] functionality."

The CASE tool is object based. Specifications for objects are stored in a central repository. The repository includes data and object definitions for the organization's business, plus all the software that comprise its systems. The tool and the associated methods constrain developers to more carefully "engineer" an information architecture.

In spite of substantial research, there exists no consensus on the economies and diseconomies of scale in software development. However, most expect that an integrated CASE tool set will offer scale economy gains for the following reasons: CASE usually enforces structure and modularization, CASE tools promote and support code reuse, productivity results from relatively bug-free machine-generated code, and CASE tolls increasingly offer automated testing and implementation facilities.

Although "developers primarily reuse their own code in the absence of incentive to search for other reuse opportunities," it was clearly recognized that "in software projects developed using CASE, the level of code reuse is perhaps the major factor that deserves attention." It was also recognized that "extensive code reuse can increase productivity by an order of magnitude and more...." However, "prior research on reuse provides little guidance as to how to construct the relevant metrics." For this study, a simple reuse metric was constructed that fit the CASE tool under examination. It involves "constructing and reusing objects ... reusable within applications and across their boundaries."

The study determined that "in a mature CASE environment" developers were more familiar with:

- how to use the CASE tool
- where reusable parts are stored
- how reusable parts can be identified and retrieved

The study focused on twenty-one financial application projects over a two-year period. "Most project personnel worked on multiple projects to broaden their experience," and "the programming language was a constant." It took a developer only weeks to become productive in the CASE tool, and only about six months for the learning curve to flattened out. Experience was figured into the statistical model.

The study examined the difference in code reuse across on-line, real-time systems and batch systems and the extent to which experience influences reuse. "The overall object reuse for the 21 projects was 2.84 times.... Average reuse for Year 2 projects was even higher at 3.95 times overall. Reuse was maximized among on-line projects at 4.11 times."

The results of the study are quite striking. They suggest not only the need for more extensive research examining reuse issues, but also more extensive research on the measurement tools and techniques of code reuse.

FURTHER EVALUATION OF THE STUDY

Studying the published document, I discovered that I was left with many questions. It appeared that many obvious factors were not considered. The study aimed to quantify CASE as directly responsible for productivity. In actuality, the study identified reuse as the overwhelming contributor to productivity, the CASE tool was the enabling technology.

Upper CASE/Lower CASE

There was no clear indication as to where in the development process — analysis, design, screen, report, code generation — CASE was applied. Although it seemed that much of the focus was in the *lower* CASE areas of screen, report, and code generation. If this in fact is true, the organization has missed a significant opportunity for additional reuse in the *upper* CASE areas of analysis and design. As the lower CASE tools become more sophisticated, less time is expended on the coding/testing effort, widening the gap between the percentage of development time spent on the front end and the back end of a system. It is intuitively understood that if analysis and design models can be reused, productivity and quality will again improve significantly. Now, all that is left is to quantify this area of reuse. Not an easy task. Information gathering as input to analysis and even the analysis work (not the modeling work) can be particularly loose and subjective and is a function more of the interpersonal and business skills of the analyst than of their technical skill.

Good parts/bad parts

A significant omission was the quality issue. No mention was made of the completeness and correctness of the systems generated through reuse. There was no indication of problems that surfaced when developers tried to reuse a part, particularly on the first instance of reuse. No matter how broad you scope, the initial creation of a part is always for a particular set of circumstance. Subsequent part reuse generally requires modification to the original. Through practical experience, we have found that unless the part is primitive it is not truly reusable until it has had from three to five separate occasions for reuse accompanied by appropriate enhancements and tweaks to accommodate each new situation. Learning how to create parts so that they are open for reuse enhancement is a learned skill. It is dependent on the quality of the analysis and design, the experience with the business and the technology, the implementation language and platform, and the corporate culture. The corporate culture is a major driving factor in reworking a part to make it reusable.

In this study, errors or lack thereof per "chunk" of system were not even mentioned, let alone quantified, nor was the amount of effort to create reusable parts quantified. The developers created a lot of stuff from reused parts, but was it the right stuff ... and what did it take for it to be the good stuff?

Teamwork

Teamwork is considered to be a major contributor to improving quality and productivity in systems development. The study considered issues of team experience in the subject matter and team experience in the technology, CASE, implementation language, and platform. The experience of the team as a team is another factor that augments quality and productivity. The ability and experience of team members in working with each other is invaluable. Knowing each other's strengths and weaknesses allows the team to expedite the development process. You learn who on the team to ask for help in a particular area. You learn what particular types of mistakes individuals tend to make. Think of your favorite sports team in a championship season. They work together like a precision machine. Everyone does what they are best at and they do that very well. The coach doesn't ask the quarterback to block and the pitcher is not expected to steal second.

Application types

The techniques embodied by this study's CASE tool is data driven. Naturally, this type of technique is more applicable to a certain class of problems. And, although certain parts of some application areas will naturally foster different level of reuse (e.g., in financial services there is a specific class of montages where one class/instantiation is only a slight exaggeration), we have found that reuse is applicable in virtually all areas and types of development. This is evidenced by the reusable class libraries that come with many language implementations, as well as, those for GUIs, graphics, and general systems functions.

Economies of scale

The economies and diseconomies of scale for object development are going to be quite different from those of traditional development. With objects, the perspective is changing from an applications focus to a larger integrated systems focus. The small granularity of objects allows for more individual independent feature enhancements to the corporate class hierarchy. The magnitude of so-called application projects will become somewhat obscured as the availability of reusable parts increases, as reusable parts are refined, and as parts management matures within an organization. Project size and structure will be much different from what we currently know.

Corporate strategy

CASE does not contribute directly to the corporate strategy, but to corporate operations that eventually will work their way through corporate tactics on to the strategy. Objects address the corporate strategy directly. Strategic object modeling is beginning to be used in large corporations as a more sophisticated technique for systems strategy and also for corporate business strategy. Unlike enterprise models using an entity relationship technique, strategic objects can be used to accurately model the business and then almost seemlessly translate through to systems planning and implementation. Very soon, CASE tools to support this effort will become critical not only to systems development, but also more importantly to general business development.

Quality

To achieve the quality and productivity of reuse, a corporate culture that advocates and incents reuse is imperative. In some organizations, writing your own string class is considered a right of passage. Just about everyone has written her or his own. Where this example might be trivial, consider a organization of fifty developers. That's fifty string classes at, conservatively, eight hours to build, test, and debug the class. Four-hundred staff-hours at, let's say, a loaded average rate of $32/hour totals $12,800 spent on string classes. You can buy entire class libraries for under $1,000. In addition there is the issue of consistency across the fifty string classes that can have a wide range of impact on applications. The corporate culture has to adapt to reward people for reusing parts, for building reused parts, and, most importantly, for adapting

continued on page 17...

≡ Your Turn ≡

How about it? How are we doing? This is the place to send us your message. Tell us what you like and what you don't, what you'd like to see but don't — or whatever other feedback you'd like to send. Send comments to Marie A. Lenzi, 225 Rector Place, NY, NY 10280; fax to THE HOTLINE, *(212)786-4032; or, if you're ordering, send along with your order.*

Please check whichever box applies:

	Read entirely	Scanned	Found helpful	Will refer to	Didn't read
Feature	❑	❑	❑	❑	❑
Standards/Certifications	❑	❑	❑	❑	❑
Getting Started	❑	❑	❑	❑	❑
Down Under	❑	❑	❑	❑	❑
Industry	❑	❑	❑	❑	❑
FYI	❑	❑	❑	❑	❑
Product Announcements	❑	❑	❑	❑	❑
Book Watch	❑	❑	❑	❑	❑

Please rate the following ***1*** (dislike or strongly disagree) to ***10*** (like or strongly agree):

Presentation of material __ Comments: _____

Accuracy of material __ Comments: _____

Overall helpfulness in your job __ Comments: _____

What else would you like to see in THE HOTLINE? _____

Any other comments? _____

_____

_____

Name, title & phone (optional): _____

hotline on OBJECT-ORIENTED *technology*

❑ ***Yes,*** plug me into the latest thinking and developments in object-oriented technology. Enter me as a subscriber at the term marked below and rush me the current issue. This is a risk-free offer — I may cancel my subscription at any time and promptly receive a refund for the unused portion.

1 year (12 issues) ❑ $249

2 years (24 issues) ❑ $478 *(save $20)*

(outside US add $30 per year for air service)

Back issues @ $25 each *($27.50 foreign)*:

Vol.1, Nos. _____

Vol.2, Nos. _____

❑ **Phone order**
Call Subscriber Services
(212)274-0640

❑ **Fax order**
Fill out term and name/address information, then fax to:
(212)274-0646

❑ **Send me a copy of** THE INTERNATIONAL OOP DIRECTORY @ $63 ($71 foreign)
(all directory orders must be prepaid; fax or mail credit card information to SIGS, 588 Broadway, #604, NYC 10012; make check payable to OOP Directory. Foreign orders must be prepaid in US dollars drawn on US bank.)

D1GC

❑ **Bill me**
❑ **Check enclosed**
❑ **Credit card orders**
❑ MasterCard
❑ Visa

Mail (and make out check, if applicable) to:
THE HOTLINE
Subscriber Services
P.O. Box 3000, Dept. HOT
Denville, NJ 07834
(foreign orders must be prepaid in US dollars drawn on a US bank)

Card# Expiration Date

Signature

Name

Company

Street/Building#

City/Province

ST/Zip/Country

Telephone

D1KC

Send me a complimentary copy of your related publication(s):

❑ THE JOURNAL OF OBJECT-ORIENTED PROGRAMMING

❑ OBJECT MAGAZINE

❑ THE C++ REPORT

Call Subscriber Services
(212)274-0640
or send order to
SIGS, 588 Broadway, #604
New York, NY 10012

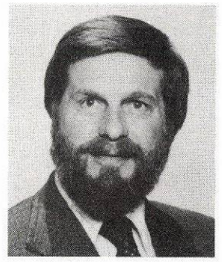
D1KC

hotline on OBJECT-ORIENTED *technology*

VOLUME 2, NUMBER 12 THE MANAGER'S SOURCE FOR TRENDS, ISSUES & STRATEGIES OCTOBER 1991

A modest survey of OOD approaches

Approaches to object-oriented design and analysis vary from silly to promising, but no one really knows how to do it yet.



David Bulman

STILL, AN OVERVIEW may help to clarify the issues involved. The two main problems in object-oriented design are deciding what objects are needed to solve a given problem and organizing them into a suitable architecture. Clearly, analysis of the requirements of the problem has to be done before the object-oriented design can be accomplished. In that sense, object-oriented design includes requirements analysis.

Is there such a thing as object-oriented analysis? No matter what language you use, you still have to determine the problem in pretty much the same way. However you get requirements — communicating with users, reading a document — and you have to find some way to understand them.

The concentration in analysis is on making sure that you can record needs and talk about them with others. Unless you're doing something for yourself, you're doing it for someone else and you have to make sure you're doing what they want you to do. Assuming you're going to implement your finished solution in an object-oriented way, is it better to communicate with users about their needs in terms of objects or in terms of functions?

The answer is, "However the users think about their problem." If they already think in terms of objects — or you can somehow get them to think about the problem that way — it would certainly be better to do your analysis in terms of objects because it is a more direct path from the problem to the code you're going to produce. If, on the other hand, you are convinced that your users think about problems in terms of the functions they want done, then perhaps there is no such thing as object-oriented analysis — for your users and you.

DIFFERENT PROBLEMS, DIFFERENT DESIGN

With the exception of language bigots, most programmers agree that different kinds of problems are best solved with different programming languages. Different problems also require different approaches for analysis and design.

continued on page 3 ...

IN THIS ISSUE

FEATURE — A modest survey of OOD approaches *David Bulman*	1	2	FROM THE EDITOR
STANDARDS/CERTIFICATIONS — What is a "certified" object programmer? *David Bellin*	6	11	INDUSTRY SIDEBAR — The reality of object management *Chris Stone*
GETTING STARTED — Perspective: investing in objects today *Adrian J. Bowles*	7	13	FYI — Industry publication excerpts
DOWN UNDER — Object oriented in Melbourne, Australia *William Haebich*	9	15	PRODUCT ANNOUNCEMENTS / NEW PARTNERINGS & ACQUISITIONS
INDUSTRY — The Object Management Group *Michael Guttman & Jason Matthews*	10	18	BOOK WATCH

From the Editor



Our feature this month is from **Dave Bulman**. Dave has been involved with object analysis and design issues much longer than most. From his experience, he offers us "A Modest Survey of OOD Approaches" and addresses the issues surrounding object analysis and design approaches.

It is clear that we cannot wait until object technology is all there before we address the issue. In "Perspective: Investing in Objects Today," **Adrian Boles** of New Science Associates offers strong arguments for planning your move to object technology. NOW!

Currently, several state legislatures are addressing the issue of certifying computer programmers. Like most things this issue has its pros and cons. On the pro side, this can help to eliminate "match book" programmers. On the con side, the people setting up the certification are not qualified to do so. **David Bellin** examines the problems and implications of this initiative in "What Is a Certified Object Programmer?"

Jason Mathews and **Michael Guttman** of **Genesis Development Corp.** offer a members' assessment of the OMG, and **Chris Stone**, the President and CEO of the **OMG** responds.

We are hearing from another member of the object community in Australia. I would like to welcome **Dr. William Haebich** of Simsion, Bowles, and Associates in Melbourne. In the "Down Under" column, he reports on the Object-Oriented Special Interest Group of the Australian Computer Society.

Currently, several of the local contract programming headhunters are combing New York for Smalltalk programmers. It seems that there's a HOT! project in town. However, on closer examination, this hot project looks more like a hot potato. The work week is twelve hours per day, six days a week. This will go on for six weeks, and the hourly rate is abominable. I don't know who the client is; I don't want to know.

So why is this project getting print space? Well, this is my opportunity to cry (scream) out against atrocities against software quality and abominations of project management. This project is doomed and is missing only one vital element to ensure absolute failure: the developers are not working the graveyard shift.

Let me articulate what dooms this project. First, all the serious Smalltalk developers have turned the job down because of the project schedule and the batch-COBOL billing rate. That leaves only Smalltalk wanna-bes as the talent base — those who have picked up a copy of Smalltalk and have taught themselves the language syntax (not object technology). This is their big opportunity to get their first real Smalltalk experience, bolster their resume, and increase their billing rate.

- Mistake 1 — major absence of concept and technology skill, experience, and talent.

Next, let's examine the schedule. A forced 72-hour work week doesn't cut it. The first week will be okay. Then it's all downhill. During the second, third, and fourth weeks, after ten hours, the brain begins to shut down: that's two hours a day wasted. After four 12-hour days, the mind drifts into a state of mild euphoria. Let's calculate: after the fourth week, the project has wasted two hours a day for the first four days of the week — 24 hours — plus two full 12-hour days — 72 hours. This gives a grand total of 96 hours per developer of wasted time. But wait, there's more. If the code wasn't bad to begin with (see mistake 1), it surely will be now. And, after the fourth week of this drill, the developers go into complete mind shutdown. That gives us two 72-hour weeks or another 144 hours per developer of wasted time. Add the 144 to the 96 hours from the first four weeks, and we come up with a total of 240 hours per developer (six weeks) of wasted time. So, in a project with a six-week duration, nearly three and one-half staff-weeks have been unproductive.

- Mistake 2 — talent burnout.
- Mistake 3 — waste of human and financial resources.
- Mistake 4 — serious product quality problems.

I can't believe responsible people permit this to occur. And yet probably every reader has been on a project like this and recognizes the consequences. But now for the best part — the finger pointing. First, the contract programmers go — they don't know what they are doing (but what about the revenues already expended?). Next, Smalltalk and object technology take it on the chin. ParcPlace or Digitalk watch out: your software is going to come flying back to you. And object technology ... Well, we gave you a try, and you just don't work. In fact, everyone and everything gets blamed, except for the real culprit — the brain surgeon who said yes to doing this project under these circumstances in the first place. The software product will not get delivered to the user. If anything is delivered, it will be bug infested and more likely than not the wrong product. The user will be furious because he has probably have been promised an application to support a new competitive line of business with a narrow market window. And the developers again will be drubbed mercilessly?

I propose an out-and-out revolt. The Soviets are standing up for their lives and their rights. Look at us. We can't even avert an obvious loser development project. How can organizations talk about the Malcolm Baldridge and continue to pull stunts like this Smalltalk project? How are we ever going to gain the respect of the user community when lemming development managers accept "challenges" like this one? You would be amazed what can be done when people ban together: put your foot down, stand tough, cliche, cliche, cliche ... JUST SAY NO!

Editor
Marie A. Lenzi

SIGS Publications, Inc. Advisory Board
Tom Atwood, *Object Design*
Grady Booch, *Rational*
George Bosworth, *Digitalk*
Brad Cox, *Information Age Consulting*
Chuck Duff, *The Whitewater Group*
Adele Goldberg, *ParcPlace Systems*
R. Jordan Kreindler, *General Electric*
Meilir Page-Jones, *Wayland Systems*
Tom Love, *Consultant*
Bertrand Meyer, *Interactive Software Engineering*
P. Michael Seashols, *Versant Object Technology*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

Hotline Editorial Board
Jim Anderson, *Digitalk, Inc.*
Larry Constantine, *Consultant*
Mary E.S. Loomis, *Versant Object Technology*
Reed Phillips, *Knowledge Systems Corp.*
Trygve Reenskaug, *Taskon A/S*
Zack Urlocker, *Borland International*
Steven Weiss, *Wayland Systems*

SIGS Publications, Inc.
Richard P. Friedman, Founder & Group Publisher

Art/Production
Elisa Varian, Production Manager
Susan Culligan, Creative Director
Kristin R. Juba, Production Editor
Caren Polner, Desktop Designer

Circulation
Diane Badway, Circulation Business Manager
Kathleen Canning, Fulfillment Manager
John Schreiber, Circulation Assistant

Administration
David Chatterpaul, Bookkeeper
Suzanne W. Dinnerstein, Conference Manager
Jennifer Fischer, Assistant to the Publisher
Laura Lea Taylor, Administrative Assistant

Margherita R. Monck, General Manager

hotline on OBJECT-ORIENTED *technology*

The Hotline on Object-Oriented Technology (ISSN #1044-4319) is published monthly by SIGS Publications, Inc., 588 Broadway, NY, NY 10012, (212)274-0640. © Copyright 1991 SIGS Publications, Inc. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the U.S. Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher. Mailed First Class. Subscription rate — one year (12 issues) $249. Foreign and Canada $279. Single copy $25.

POSTMASTER: Send address changes & subscription orders to The Hotline, Subscriber Services, P.O. Box 3000, Dept HOT, Denville, NJ 07834.

Submit editorial correspondence to Marie A. Lenzi, 225 Rector Place, Suite 12N, New York, NY 10280. voice: (212)945-9447, fax: (212)786-4032.

SIGS PUBLICATIONS

Publishers of *Hotline on Object-Oriented Technology, Journal of Object-Oriented Programming, Object Magazine, The X Journal, The C++ Report, The Smalltalk Report,* and *The International OOP Directory.*

Back issues available

Vol.2, No. 11/September '91 ≡ From applications to frameworks • *Urlocker* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Getting started with object technology: efffectively planning for change • *Lorenz* ≡ Object statistics on the way • *Lenzi* ≡ On objects and bullets • *Onart*

Vol.2, No. 10/August '91 ≡ Distributed object management: improving worker productivity• *Osher* ≡ Getting the best from objects: the experience of HP • *Coleman & Hayes* ≡ Applications: EC employs object technology ... • *Stein* ≡ Capacity Planning: Fiddling while ROMs burn • *Rovira*

Vol.2, No. 9/July '91 ≡ Multimedia is everywhere! • *Weiss & Berman* ≡ Developing an object technology prototype • *Kulmek* ≡ Object-oriented capacity planning • *Rovira* ≡ How OOP has changed our developmental lifecycle • *Zeik* ≡ Modularization of the compurer system • *Angenius*

Vol.2, No. 8/June '91 ≡ Domain of objects: the Object Request Broker • *Dyson* ≡ Object-based approach to user documentation • *Reeder* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Do we need object-oriented design metrics? • *Hopkins*

Vol.2, No.7/May '91 ≡ Hybrid object-oriented/functional decomposition for software engineering • *Henderson-Sellers* ≡ So, what makes object databases different? (Part 4) • *Blakey* ≡ Using the generic application to solve similar domain problems • *Gossum* ≡ Experiences using CLOS • *Hopkins* ≡ International Conference on Object-Oriented Technology, Singapore • *Bennett*

Vol.2, No.6/Apr. '91 ≡ An artist's perspective of programming • *Rovira* ≡ So, what makes object databases different? (Part 3) • *Blakey* ≡ Moving from Pascal to C++, Part 3 • *Gole* ≡ Object projects: what can go wrong • *Stewart* ≡ Reflections from LOOK-'91 • *Anders-Angenius*

Vol.2, No.5/Mar. '91 ≡ Large-scale object systems development • *Morgan & Davis* ≡ Data definition languages • *Blakey* ≡ Object disorientation • *Kerr* ≡ Moving from Pascal to C++, Part 2 • *Keller* ≡ The Object Management Group • *Barber*

Vol.2, No.4/Feb. '91 ≡ Moving from Pascal to C++ • *Sollisch* ≡ The Australian object-oriented scene • *Henderson Sellers* ≡ So, what makes object databases different? • *Blakey* ≡ Managing the evolution • *Novobilski* ≡ 1990: the year of realism • *Love* ≡ Choosing a configuration for object development • *Moody*

Vol.2, No.3/Jan. '91 Special issue: CASE and Methodology ≡ Object orientation: stop, look and listen! • *Page-Jones* ≡ The spiral model for object software development • *Wasserman & Pircher* ≡ Bootstrapping object-oriented CASE • *Wybolt* ≡ A CASE tool for object software development • *Ackroyd* ≡ Comments on "Race for the database pennant" • *Loomis*

Vol.2, No.2/Dec. '90 ≡ Waiting for OODB2 • *Durham* ≡ Object modeling • *Singh* ≡ Object-oriented COBOL — the next generation • *Adams & Lenkov* ≡ Object-oriented training in the UK • *Hopkins*

Vol.2, No.1/Nov. '90 ≡ Objects by teamwork • *Constantine* ≡ Race for the database pennant: a third-generation manifesto • *Stein* ≡ Project team descriptions *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.12/Oct. '90 ≡ Crown jewels: the value of OOT in MIS departments • *Bonar* ≡ Why is an object database faster than a relational system? • *Loomis* ≡ Visual programming object language environments • *Klausner* ≡ Hype — necessarily (or a necessary) evil? • *Adams* ≡ On Managing: Déjà Vu • *Lenzi*

Vol.1, No.11/Sep. '90 ≡ Introducing OOPS through design — it's all in the words • *Copen Goldstein* ≡ The Australian object-oriented scene • *Henderson-Sellers* ≡ Objects on tour (Review of Borland's OOP World Tour) • *Lenzi* ≡ Design principles for modular product development • *Bennett* ≡ Asset management and object-oriented technology • *Hopkins & Warboys*

Vol.1, No.10/Aug. '90 ≡ WyCASH+: An application built within an OOP environment • *McCullough & Deshler* ≡ Mathematics and methodology • *Martin* ≡ Organizing for reuse • *Goldberg & Rubin* ≡ CALS Expo–89 Conference • *Purdy*

Vol.1, No.9/July '90 ≡ Cognos and Eiffel: a cautionary tale • *Leathers* ≡ What's in an object anyway? • *Novobilski* ≡ The impact of Windows 3.0 on the OOP market • *Urlocker* ≡ Introducing reuse into the development process • *Goldberg & Rubin*

Vol.1, No.8/June '90 ≡ A taste of the USENIX C++ Conference • *Wells* ≡ Comparing relational and object database management systems part 2 — Composites, flexibility, and extensibility • *Martin* ≡ Case studies in prototyping, part 3 • *Goldberg & Rubin* ≡ Object-oriented requirements anilysis • *Berard*

Vol.1, No.7/May '90 ≡ Object-oriented development: not business as usual • *Lorenz* ≡ Comparing relational and object database management systems part 1 — Single paradigm programming is the key difference • *Martin* ≡ Case studies in prototyping, part 2 • *Goldberg & Rubin* ≡ Understanding the recursive/parallel lifecycle • *Berard*

Vol.1, No.6/April '90 ≡ Lifecycle approaches • *Berard* ≡ A data manipulation language for C++ developers — a proposal • *Atwood* ≡ Case studies in prototyping, part 1 • *Goldberg & Rubin* ≡ I already learned in kindergarten all I needed to know about object-oriented design • *Ramakrishnan*

Vol.1, No.5/March '90 ≡ Objects of desire • *Stone* ≡ OMG: The state of the object-oriented DBMS market • *Atwood* ≡ Talking to project managers • *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.4/Feb. '90 ≡ Greed: a complete example • *Love* ≡ OMG: an industry push for standards • *Atwood* ≡ Talking to project managers • *Goldberg & Rubin* ≡ OOP in the 1990s • *Urlocker*

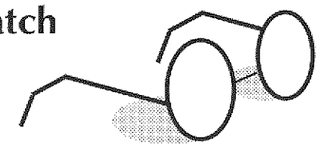
To order, see back page; for reprints of individual articles, call Michael Biggerstaff at Reprint Management Services (717) 560-2001.

Calendar

The Hotline Calendar presents conferences and meetings that focus exclusively on object-oriented technology. To have a meeting or conference listed, please send the dates, conference name and location, sponsor(s) and contact name and telephone number to the Editor: Marie A. Lenzi, 225 Rector Place, Suite 12N, New York, NY 10280, fax (212) 786-4032.

October 28–31 **International Working Conference on the Object- Oriented Approach in Information Systems** Quebec City, Canada Sponsor: IFIP Contact: Bernard Moulin, Fax (418) 656-2324	**October 28–31** **SCOOP-Europe** Westminister, London, UK Sponsor: *JOOP*, *Object Magazine*, & Wang Institute of Boston University Contact: Suzanne W. Dinnerstein, (212) 274-0640	**November 5–7** **Database World** Chicago, IL Sponsor: DCI Contact: Customer Service, (508) 470-3880	**November 18–22** **C++ at Work** Santa Clara, CA Sponsor: *The C++ Report* & Wang Institute of Boston University Contact: Suzanne W. Dinnerstein, (212) 274-0640	**November 21–22** **Object-Oriented Analysis and Design Seminar** Orlando, FL Sponsor: DCI Contact: Customer Service (508) 470-3880

Book Watch



The "Book Watch" column does not contain book reviews. These listings are abstacted from press releases provided by the publishers, and no endorsement is implied. Please send announcements to the Editor: Marie Lenzi, 225 Rector Place, Suite 12N, New York, NY 10280, fax (212) 786-4032.

Visual Basic Programming with Windows Applications

Douglas Herbert; Bantam Books, New York $44.95

Bantam Electronic Publishing, a division of Bantam Books, is currently shipping Visual Basic Programming with Windows Applications by Douglas Herbert. With this guide, Visual Basic users can create powerful programs that have the same look and feel as other Windows applications. The book includes a disk that features sample applications and utilities created with Visual Basic. These will illustrate Visual Basic's most important concepts and help you develop Windows applications and create executable application files that can be run directly from the Windows environment.

The Standard C Library

P. J. Plauger; Prentice Hall, Englewood Cliffs $28.00 (soft cover)/$38.00 (hard cover)

Prentice Hall announced the publication of The Standard C Library by P. J.. Plauger, a comprehensive treatment of ANSI and ISO standards for the C library. The Standard C Library features practical advice on using all fifteen headers in the standard C library. Each chapter is devoted to an individual section of the library. Complete code for the ANSI standard C library is included in the book. In addition to detailing the C library, the book focuses on the design and implementation of libraries (library building). Plauger discusses the concepts, design issues, and trade-offs, associated with library building. The book also covers internationalization issues and writing locale-independent programs.

Microsoft Windows 3: Finding the Cracks

David Kahn; D. Lawton Associates, Natick, MA $99.95

D. Lawton Associates has announced the publication of Microsoft Windows 3: Finding the Cracks. Written to give realistic information about Windows, the author, David Kahn, helps readers evaluate the product, not the hype, before their time and money are committed to Windows.

Eiffel: The Language

Bertrand Meyer; Prentice Hall, Englewood Cliffs, NJ $30.80

Interactive Software Engineering Inc. has announced the publication of Eiffel: The Language by Bertrand Meyer, President of ISE and main designer of the Eiffel object-oriented language. This is one of the first entries in the newly announced Prentice Hall Object-Oriented Series, a joint undertaking of Prentice Hall USA and Prentice Hall International, for which Dr. Meyer serves as Series Editor. Eiffel: The Language combines a pleasant and pedagogic style with a high degree of precision and formality. As such, it is suitable for novices as well as advanced users and authors of Eiffel compilers and other tools.

C++ Components and Algorithms

Scott Robert Ladd; M&T Books, Redwood City, CA $39.95

C++ Components and Algorithms is a comprehensive reference for experienced C programmers who develop and maintain powerful C++ applications. This book covers such topics as memory management, indexed files using B-Trees, mathematical programming, and adaptive algorithms. The in-depth discussions and numerous source code examples are geared toward an understanding of C++'s inner workings. The programs and classes presented are compatible with AT&T, Borland, Microsoft, and Zortech C++, making them valuable to a wide audience of C programmers. All of the source code is available on disk in PC/MS-DOS format.

Clean Coding in Turbo Pascal 6

Amrik Dhillon; M&T Books, Redwood City, CA $36.95

Clean Coding in Turbo Pascal 6 is a definitive source of information for Borland Turbo Pascal programmers. It contains in-depth coverage of Turbo Vision and Object Windows, Turbo Pascal's object-oriented libraries, plus information on Turbo Pascal for Windows. Topics covered include events, windows and dialog boxes, input lines, data transfer, color palettes, streams and resources, automatic message response methods, parent and child windows, modeled and unmodeled dialog boxes, and dynamic link libraries.

Advanced Graphics Programming in C and C++

Roger T. Stevens; M&T Books, Redwood City, CA $39.95

Advanced Graphics Programming in C and C++ by Roger T. Stevens details the fundamentals of graphics programming for the IBM PC and compatibles, teaching C and C++ programmers how to create compelling graphic images easily and efficiently. Because it contains all of the source code, readers can easily modify the functions to suit their specific need. All listings are available on disk in MS/PC-DOS format.

... continued from page 1

Intuition

The first method object-oriented programming enthusiasts often propose for object-oriented design goes like this: first, decide which parts of the solution are to be objects.

The problem is, of course, how? Use a browser for several days? Meditate for a week? If you attend OOPSLA conferences you'll find that probably 80% of the people there say, "Well, the objects are obvious." And they are, if you're brilliant or if it's a tiny problem. But, if you're like most of us and it's a large problem, the objects are not obvious.

A valuable book, *Object-Oriented Analysis* by Shlaer and Mellor[1] was published in 1988. In Section 3.2, "Identifying Objects," they said, "Identifying objects is pretty easy to do ... ask yourself, 'What are the things in this problem?'"

The Class Library Approach

Many workers using a programming environment with a large class library say that most needed classes can be found in the library.[2, 3] This approach certainly works extremely well if your class library is well suited to your problem. If a large portion of the difficult programming has to do with the user interface, then something like TurboVision, MacApp, or the Smalltalk class library can solve most of your problem.

There do not, however, seem to be any class libraries well suited to the problem of, say, process control for a candy factory. Such libraries will probably be available sometime in the future, but not now.

What About Inheritance?

When your library fails you, people generally say, "Inheritance will save me." Very often, you'll try to find some class in the library from which you can inherit the class you actually need (with suitable modifications). The numbers of papers and books espousing this idea are too numerous to mention. If you're a novice object-oriented programmer, you probably will consider this the principal method of object-oriented design.

Unfortunately, the same limitation applies here as with normal dependence on a class library: the library must be well suited to the problem at hand.

In addition, recognizing the opportunity for inheritance seems to work well primarily when a small, highly capable team is working on a familiar problem that is small enough to be completed on schedule. Otherwise, most opportunities for using inheritance seem to be discovered only after the programmer has implemented a number of objects and recognizes that the commonality between them could be abstracted into a higher-level class. As a result, I think that one ought to consider inheritance more an implementation tool than a design tool.

Nouns and Verbs

The next design method is the one that most programmers have heard of. Usually called "nouns and verbs," it's also sometimes called the "paragraph" method. It holds that you should write a paragraph in English (or any natural language) that describes the problem. Then, you should look for the objects/classes among the noun phrases in this description. Extreme proponents suggest underlining the nouns in one color and the verbs in another. Some say the verb phrases in the description will represent the operations on the objects. Although this method, invented by Russ Abbott,[4] was popularized by Grady Booch,[5] Booch doesn't think this method actually scales up to industrial-size problems.[6] The reason he described it is that it's an excellent way to begin to learn about objects, thus it's something all beginners should think about.

> ***The two main problems in object-oriented design are deciding what objects are needed to solve a given problem and organizing them into a suitable architecture.***

At one large company, six people spent over a month creating a paragraph they could all agree on. Why did it take so long? If you remember Physics 101 lab, you probably remember the frustration of trying to make measurements taken with inadequate instruments produce predicted results. Some of you even (like me) realized that the best approach really was to work backward from the expected result to find out the measurement you were supposed to observe.

I believe that exactly the same thing was going on with that six-person team. They had already decided — through meditation or whatever — what the proper objects would be for this problem. Since they already "knew" pretty well what the objects were supposed to be, they were trying to write a paragraph such that when they underlined the nouns, those would be the objects they were trying to get. I've seen this happen. Wherever, as a consultant, I develop an object-oriented design, there's someone who favors the paragraph method. He'll say, "I can do it just as well with the paragraph method." Then, he will take my solution and write a paragraph

describing it, and — lo and behold! — if you underline the nouns in his paragraph they're the right objects for the problem. That's an interesting, and possibly valuable, method of documentation. It is not, however, a design method.

I am devoting so much space to the use of nouns and verbs because several of the other approaches described here are little more than elaborate notational schemes. While some are well developed, the actual method associated with several of them boils down to examining a description of the problem for the important things mentioned in the requirements. Thus, although the pure nouns and verbs method has been fairly completely discredited, traces of it appear in several other methods. Nouns are still useful because it's certainly true that most objects are going to be nouns. But, having found obvious objects from nouns, usually only vague guidance is available for the remaining objects. Since you can make any verb into a noun, the objects derived by the noun and verb approach are also likely to depend on the writing style of the requirements analyst. Finally, most of the verbs in the description are probably not really sources of operations on the objects at all. It is far more likely that the verbs will represent relationships between entities in the information model.

Finding Entities in the Problem

Bertrand Meyer, in *Object-Oriented Software Construction*,[7] says, "Perhaps the most useful technique for finding classes ... is to look for meaningful, external objects. Many classes just describe the behavior of objects from the abstract or concrete reality being modeled ...This idea often yields the fundamental classes of a system, obtained directly from their external counterparts."

Many very capable object-oriented designers approach problems in just this way. Meyer has described this approach more articulately than most authors. It is certainly a suitable basis for deriving many of the objects in small problems and some of the objects in more complex problems.

Information Modeling

Information modeling has traditionally been the domain of MIS people. Most people doing embedded systems, systems programming, or real-time programming think that information modeling doesn't have anything to do with them. But there's little doubt that what finally emerges as the way to do object-oriented design is going to depend heavily on information modeling, no matter what the domain of your problem.

Several of the best-regarded object-oriented design methods today are variations on the information modeling theme.[1,8–10] Some use fairly classic entity relationship diagrams, while others introduce object diagrams that show objects, their relationships, and their operations.

Unfortunately, every author who has written about this has pushed aside the big question: they don't say how to find the entities, beyond telling you to look for the nouns (sound familiar?) in either the user's problem description or the requirements statement. If nothing else, though, drawing object diagrams at least focuses your attention on the idea that you want to be thinking about objects rather than functions or processes. To see how to actually do information modeling, it is useful to go to books dealing specifically with the subject.[11]

Object Modeling Technique

A more recent technique, object modeling can be said to be the object-oriented design rendering of information modeling. The authors[10] have partitioned their model into three viewpoints.

Together, the object, dynamic, and functional models show how the objects change over time, the different states that they can take on, and the functions that each one — or the whole system — can perform.

The first step in this method is the same as in most of the other methods: identify the objects or classes. Their advice is, "Begin by listing candidate object classes found in the written description of the problem. Don't be too selective. Write down every class that comes to mind. Classes often correspond to nouns."

In other words, they haven't really said how to determine what the objects are, but they have done quite a nice job of showing how to go about documenting your thinking as you develop object-oriented designs.

Data Flow Diagrams

The question of whether or not a data flow diagram (DFD) is useful for object-oriented design has attained almost the same religious status as which text editor or programming language is best. Many people state that DFDs are really performing functional decomposition and that, being functionally oriented, they are a bad choice for object-oriented design. Perhaps the most important thing to note about DFDs is that they actually have little to do with functional decomposition. They describe the flow of data in a system, where the data is stored, and how the data is communicated to and from the outside world. What some people refer to as the "functions" in a DFD were described by their originator as simply "rules for transforming the input data to the output data."

In particular, DFDs can be used to find objects that represent stores of data describing things in the problem domain. They can also be used to find objects that encapsulate interfaces between the system and the outside world. Descriptions of the operations for the objects are found in the data transforms that access the objects in the DFDs.

A newer kind of structured analysis, called essential

New Partnerings and Acquisitions, October 1991

Some of the world's largest computer vendors have announced support for **SunSoft's** Solaris on the Intel platform including **Novell, ASCII Corp.** (Japan's largest PC and UNIX software company), **Dell Computer Corp., AST Research, Toshiba, CompuAdd,** and **NetFRAME.** Intel has announced plans to cooperate with SunSoft for greatest optimization on the X86 platform. **CompuAdd, Solbourne, Tatung, Toshiba, ICL,** and **Sun Microsystems Computer Corp.** have announced their endorsement of Solaris for their SPARC products. More than fifty software developers, such as **Lotus, Ashton-Tate, WordPerfect, Oracle, Cadence Ventura Software, Informix,** and **ASK/Ingres** also announced support for Solaris.

UniSQL has announced the formation of the UniSQL Technology Partners program. **Massachusetts Institute of Technology's** Intelligent Engineering Systems Laboratory and NTT Data Communication Systems Corporation are the first UniSQL Technology Partners Program participants. The program is designed to assist organizations that rely heavily upon the use of sophisticated data and information management technologies for their business success. The Program offers participants early access to UniSQL products, including (1) the UniSQL/X next-generation database management system, the unification of relational and object-oriented data models into a next-generation DBMS; and (2) UniSQL/M multidatabase management system, a commercial multidatabase management system designed to provide a seamless development view of multiple heterogeneous database systems. Currently under development is UniSQL/4GE, a GUI-based, next-generation application development environment designed to support object-oriented, multimedia-based application development efforts.

Interactive Development Environments, Inc. (IDE) has announced its support for Solaris 2.0, **SunSoft's** UNIX-based system software environment. Sunsoft is a subsidiary of **Sun Microsystems**. IDE's SOFTWARE THROUGH PICTURES integrated CASE environment has been available on every platform ever produced by Sun. IDE plans to move SOFTWARE THROUGH PICTURES to SunOS 5.0 and OpenWindows Version 3.0, which are components of Solaris. IDE is also redesigning the user interface of SOFTWARE THROUGH PICTURES to comply with Open Look.

Seven months after leaving as head of **Mentor Graphics Corp.'s** Silicon Design division, James Hammock has joined the board or directors at **Objectivity Inc**. Hammock has been President and Chief Executive Officer of Silicon Compiler Systems before that company merged with Mentor. He also served as Chairman of the Electronic Design Automation Companies committee. In May, Mentor and Objectivity jointly announced compatible CAD framework initiative integration toolkits.

The **Eiffel** object-oriented language and environment have been selected as the primary software development platform by the **European** EC's ESPRIT II "Business Class" project. This three-year, 640 man-month effort has been placed under the responsibility of a consortium of ten European companies in four different countries. The project will result in object-oriented CASE tools addressing the needs of analysis, design, implementation, and maintenance of business applications. It will also focus on methodological issues and on the integration of object-oriented technology and Eiffel with existing software products, particularly in the area of databases and user interfaces.

The Board of Directors of **Instantiations, Inc.** has announced that Michael W. Taylor has been elected President and CEO of the company. Mr. Taylor is a principal in the company and most recently held the position of Vice President and Chief Operating Officer. He has been with Instantiations for most of its four-year history.

KnowledgeWare, Inc. and **IntelliCorp, Inc.** have jointly announced that the boards of directors of both companies have approved a definitive merger agreement under which KnowledgeWare will acquire IntelliCorp. Under the merger agreement, KnowledgeWare will issue .13402 share of its common stock in a tax-free exchange for each outstanding share and option of IntelliCorp common stock. The merger, which is subject to the approval of IntelliCorp's shareholders and other conditions, will be accounted for by KnowledgeWare as a pooling interest. The merger is expected to be consummated in late October or early November.

C++ and Application Frameworks/Turbo C++ and Turbo Vision

Borland International, Inc.
1800 Green Hills Rd.
PO Box 660001
Scotts Valley, CA 95067
(408) 439-4825

Borland is now shipping **Borland C++ and Application Frameworks.** The new application frameworks included in the product—ObjectWindows and Turbo Vision—simplify application development by providing an object-oriented foundation on which to build software. The product also features both a 100 percent ANSI C compiler and a C++ compiler, Microsoft Windows 3.0 support, turbo drive programmer's platform and compiler, turbo debugger, turbo profiler, and turbo assembler.

Being shipped by Borland is **Turbo C++ and Turbo Vision**, a programming language product comprised of Borland's Turbo C++ compiler and Turbo Vision for C++. Including two compilers in one—an ANSI C compiler and a C++ compiler conforming to AT&T C++ 2.0 specification—Turbo C++ and Turbo Vision also contain a complete C++ application framework, a programmer's platform, VROOMM overlay manager, and turbo help. It is compatible with Borland C++ so that code written using Turbo C++ can be ported to Borland C++.

KnowledgeMan 3.0/GURU 3.0

Micro Data Base Systems, Inc. Two Executive Dr.
PO Box 6089
Lafayette, IN 47903
(317) 447-1122 x290

Micro Data Base Systems, Inc. is now shipping **Version 3.0 of both KnowledgeMan and GURU**. KnowledgeMan is a relational database management system for business applications. GURU is a comprehensive expert system environment. Version 3.0 allows developers experienced in procedure-oriented programming to move beyond relational architectures to incorporate object-based elements into their applications. This transition is accomplished through the enhancement of KnowledgeMan and GURU's programming language (KGL), and the introduction of binary large objects (BLOBs), user-defined functions (UDFs), and a compiler. Utilizing these four features, multimedia presentations can be produced with just one interactive SQL SELECT. KnowledgeMan and GURU are both available for single-user MS DOS-based PCs, OS/2, most popular LANs, DEC VAX/VMS, and SUN UNIX environments.

MEIJIN++ 1.1

Network Integrated Services, Inc.
221 W. Dyer Rd.
Santa Ana, CA 92707
(714) 755-0995

Network Integrated Services, Inc. is now shipping MEIJIN++ Version 1.1—a 109-class library that allows programmers to use the finest features of the C++ language to develop mathematical, statistical, and queuing models efficiently. MEIJIN++ allows developers to reduce complex models to a collection of interacting entities configurable at runtime. MEIJIN++ includes a 600-page user's manual, object code libraries for Borland and Zortech compilers under MS-DOS, and documented, portable C++ 2.1-compliant source code.

Solaris 1.0/ Solaris 2.0

SunSoft
2550 Garcia Ave.
Mountain View, CA 94043
(415) 336-0678

SunSoft, the system software subsidiary of Sun Microsystems, Inc., has introduced two versions of Solaris. **Solaris 1.0** is based on SunOS 4.1.1, OpenWindows Version 2, and DeskSet Version 2. It is available immediately for all SPARC vendors in a shrink-wrapped package that includes a compact disc and documentation. **Solaris 2.0** is based on SunSoft's latest SVR4-based SunOS 5.0 with symmetric multiprocessing and multithreading and enhanced ONC. It features OpenWindows Version 3 and DeskSet Version 3. Solaris 2.0 is offered on the SPARC and INTEL platforms. It maintains source compatibility with Solaris 1.0, providing a smooth migration path for software developers to SVR4.

Eiffel on the NeXT/ArchiText

Interactive Software Engineering, Inc.
270 Storke Rd., Ste. 7
Goleta, CA 93117
(805) 685-1006

Interactive Software Engineering and Quarried Software have announced that Eiffel is now available on the NeXT platform. **Eiffel on the NeXT** includes (1) the complete Eiffel distribution from ISSE, including the development tools and the complete set of class libraries as documented in *Eiffel: The Libraries* (the X Window System-based graphics libraries are included); (2) a set of graphics library classes that duplicate the X graphics classes, but are implemented using NextStep and Display PostScript; (3) a set of Eiffel classes corresponding to the NeXT AppKit Objective-C classes; and (4) online documentation (using the Digital Librarian) for all classes within the Eiffel class library. An important complement to the environment is the Eiffel Shelf, a selection of object-oriented modules contributed by third parties.

Interactive Software Engineering also announced the release of **ArchiText**, the language-driven document constructor designed to write and maintain structured documents. Produced using purely object-oriented programming techniques, ArchiText features a graphical user interface, based on the Motif GUI standard, that makes it quick and easy to manipulate complex structures. Structures can be built both from system-generated templates or by typing partial or complete phrases that are then parsed and completed by ArchiText's sophisticated built-in parser. A unique feature of ArchiText is its ease of adaptation to any context-free language. ArchiText also provides semantic facilities for executing programs, even not completely refined ones, for software prototyping and program testing.

RELATE

ObjectSoft Corporation
678 Alexander Ct.
River Vale, NJ 07675
(201) 930-0582

ObjectSoft's first product, **RELATE**, incorporates a three-dimensional, electronic rolodex PC program. Users can "soft staple" any number of documents to any entry in the electronic phone book. In RELATE, the relationship is formed with a soft link that makes no change in the original document. Since the soft link itself is an object, multiple users can reference the same link, and links can be made by one or more users with the some original document. RELATE augments the new object linking and embedding (OLE) technology. Having retrieved an object with RELATE, the user can take advantage of OLE to make compound documents. RELATE is comfortable in Windows standard or enhanced modes, and requires no additional hardware or memory. Network support is available for Netware and NetBios LANs, including Novell.

systems analysis, may be especially suited to object-oriented design because it concentrates on the behaviors that have to be produced when the system receives stimuli. Most behaviors comprise two parts: producing a required response and changing the internal state of the object. When we begin to examine data stores, and the changes necessary to produce required responses to various events, we see that these data stores must be part of the objects. If we're supposed to produce the sum of all measurements in a certain amount of time, we must have stored those measurements somewhere; so we have a data store. Now we know we need to build an object around that data store.

Responsibilities

A relatively new approach to object-oriented design could be called "responsibility-driven" design.[12] Basically, its contribution is to use an intermediate step to help ease the task of defining object operations, although not objects themselves. First, describe the responsibilities of each object, without being too detailed in the description (not worrying about necessary parameters, for example). Later, derive the required operations from the responsibilities. This has the advantage of dividing one large step into two smaller ones, making the task easier to conquer.

SUMMARY

In object-oriented design, intuition isn't sufficient. Objects are only obvious for very small problems. Of course, it's true that over half of the objects in your solution will be obvious the first time you look carefully at the problem. The trouble is, it's the last 20% of the objects you have to find that will take the last 90% of your time. A huge class library is not going to solve your problem, and a browser won't do it either. If we ever get a huge class library of truly reusable classes, what we will have then is a large data retrieval problem instead of an object discovery problem.

Inheritance is certainly one of the great things about OOP, but it's not the solution. In some cases, you can ignore it until you begin implementation. Nouns and verbs won't help whenever you need help. They work fine on homework problems; but, on a real problem, nouns and verbs will kill you.

Don't be prejudiced against entity relationship diagrams: they really do help you think about entities and, thus, about objects. Whatever method you use should be based on some model of the problem, not just the solution. Model the user's world; find the objects the user considers important. When you finish a design, no matter how you model it, some of it will be objects that come from modeling the solution — hash-coded indexes, for example, (certainly not something any user would request). But most of the objects will either be objects in a model of the problem or objects in a model of the user's world. ≡≡

References

1. Shlaer, S. and S. J. Mellor *Object-Oriented Systems Analysis — Modeling the World in Data*, Yourdon Press, Englewood Cliffs, NJ, 1988.
2. Goldberg, A. *Smalltalk-80: The Interactive Programming Environment*, Addison-Wesley, Reading, MA, 1984.
3. Schmucker, K. J. *Object-Oriented Programming for the Macintosh*, Hayden Book Co., Hasbrouck Heights, NJ, 1986.
4. Abbott, R. J. *Report on Teaching Ada (Revised)*, Science Applications, Inc., 1980.
5. Booch, G. *Software Engineering with Ada, 1st ed.*, Benjamin/Cummings Publishing Co., Menlo Park, CA, 1983.
6. Booch, G. *Software Engineering with Ada, 2nd ed.*, Benjamin/Cummings Publishing Co., Menlo Park, CA, 1987.
7. Meyer, B. *Object-Oriented Software Construction*. Prentice Hall, Englewood Cliffs, NJ, 1988.
8. Coad, P. and E. Yourdon. *Object-Oriented Analysis, 1st ed.*, Yourdon Press, Englewood Cliffs, NJ, 1990.
9. Booch, G. *Object-Oriented Design with Applications*, Benjamin/Cummings Publishing Co., Redwood City, CA, 1991.
10. Fleming, C. C. and B. von Halle. *Handbook of Relational Database Design*, Addison-Wesley, Reading, MA, 1989.
11. Rumbaugh, J., et al. *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, NJ, 1991.
12. Wirfs-Brock, R., B. Wilkerson and L. Wiener. *Designing Object-Oriented Software*, Prentice Hall, Englewood Cliffs, NJ, 1990.

David Bulman first started trying to develop a method for OOA&D in 1983. He began to teach a five-day seminar on the subject in 1986, and OOA&D has been his primary vocation and avocation since then. In 1975, he founded Pragmatics, Inc. David can be reached by phone at (808) 883-9011 or on CompuServe as 73147,1376.

This article is based on a talk given by the author at Borland International's recent Borland Languages Conference. *Developer's Insight* [(617) 740-0135] was the source for this article and holds the copyright.

STANDARDS/CERTIFICATIONS ≡≡

What is a "certified" object programmer?

by David Bellin

This column is intended to debate the national policy implications of our technology.

It brings debates out of the halls of government and into the object community. Many of these issues are controversial, and your reaction to and debate with my ideas is encouraged. Most of all, I hope to spur you to interact with elected officials and community organizers.

This month, I'd like to raise the issue of professional certification. In the object world, this may fast become a significant issue. Object programming, especially in the "pure" languages, is just coming into its own. And so are complaints about practitioners who just don't know what there doing or who present themselves as more experienced than they in fact are. I have already heard of disappointed attendees at training courses that charge fees some might consider almost extortionate.

Such problems might be expected in a new field. However, we'd do well to recognize that object technology did not spring up overnight. It is an outgrowth of the work done for some decades now — installing working computer systems in academia and industry. We are part of the data processing/computer science profession. And a large consortium known as the Institute for the Certification of Computer Professionals (ICCP) (Des Plaines, IL; (708) 299-4227) has been pushing the idea of formal qualification proceedings. For some years, they have been administering exams that lead toward the Certificate in Data Processing, the Certificate in Computer Programming, and the Certificate in Systems Analysis.

The ICCP is no light-weight outfit. It's a consortium of many the major organizations in the field: ACM, IEEE, and DPMA are just three of about a dozen involved. Now, the certification process is voluntary, and most employers don't require the certificates or the continuing education they imply. In fact, many employers don't even know they exist!

So why bother? Licensing and certification are coming regardless of our wishes. Currently, legislation is in the works in New Jersey, New Mexico, Pennsylvania, and California. And, newspapers have mentioned pending legislation in Texas, Ohio, and Tennessee, although I have not been able to confirm it. Assemblywoman Barbara Kalik (Democrat-NJ) gives the standard support: "If I own a business and go to a software designer, how do I know that he knows what he says he knows? It's an infant industry, at least as it affects the public, and there are certain minimum standards that anyone hanging out a shingle as a software designer should adhere to."

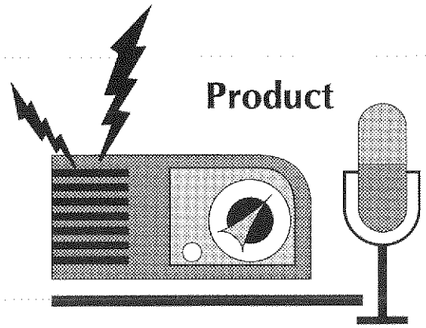
The New Jersey bill would require passing examinations (content to be determined) and payment of a biannual licensing fee (amount to be determined). This program would be administered by an appointed (by the Governor) State Board of Software Designers. While the ICCP stated to me that they would like to be part of the testing process, there is no assurance they will be.

From the employees' and consultants' viewpoints, it might be worthwhile to have another piece of paper affirming their knowledge. I hold the CDP myself. And, if you fail, nobody will ever know! For managers, a certificate holder has a known base of knowledge from which they can start.

If our profession doesn't clean up its own act, the government will step in and force us to. Licensing or certification is regulated for diverse professions already, including pilots, nurses, engineers, barbers, doctors, accountants, and lawyers. Why shouldn't the people who build the systems that make possible daily life in the modern age be subject to similar controls?

A recent ICCP survey, which should be made public by the time you read this column, shows an even split among computer professionals on licensing. Let me know what you think, and we can pursue the subject in coming issues. ≡≡

David Bellin is with Pratt Institute, New York. dbellin@pratt.bitnet.



Product Announcements

Product Announcements is a service to the readers of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY; it is neither a recommendation nor an endorsement of any product discussed.

Objectworks/C++ Release 2.4
ParcPlace Systems
1550 Plymouth St.
Mountain View, CA 94043
(415) 691-6700

ParcPlace Systems' new release of **Release 2.4 of Objectworks/C++** allows programmers to use existing makefiles without any modification. It now integrates easily with UNIX tools that are used in development of complex software systems, including source code control systems (RCS, SCCS), external editors (emacs and vi), and build utilities (make, imke, gmake, and nmake). This integration is completely nonintrusive; the open interface allows Objectworks/C++ to work with new UNIX development tools such as yacc++ and lex++. Ideally suited for use on large C++ projects, Objectworks/C++ scales as projects expand and increase in complexity. Additional new features of Objectworks/C++, Release 2.4 include debugger enhancements for peer and light-weight processes support. In addition, it now supports Sun's Light Weight process Library, which allows programmers to work on multiple tasks simultaneously. Objectworks/C++ is well suited for technical, engineering, and scientific professionals developing C++ software in a UNIX environment. It is ideal for programming teams working on medium to large C++ projects.

Kanji support for GemStone
Servio Corporation
1420 Harbor Bay Pkwy.
Alameda, CA 94501
(415) 748-6200

Servio Corporation has announced the shipment of a **Kanji object database management system** (ODBMS). Servio's GemStone now supports manipulation of extended UNIX code (EUC) standard Japanese character strings. Kanji support is immediately available in Japan and will be made available worldwide this fall. The initial Kanji release supports the Japanese Industrial Standards (JIS) character set, using the EUC representation. GemStone Kanji capabilities provide support for storage, retrieval, indexing, concatenation, and all other functions normally associated with ASCII string manipulation. By supporting full indexing of Kanji character strings, GemStone assures high performance for applications handling complex multimedia data including Kanji text. In addition to Kanji capabilities, Servio will Japanize GemStone documentation, the developer interface, and other product features.

TurboCASE 4.0
StructSoft, Inc.
5416 156th Ave. S.E.
Bellevue, WA 98006
(206) 644-9834

StructSoft has scheduled shipment of **Version 4.0 of TurboCASE** for the fourth quarter of 1991. TurboCASE 4.0 is an integrated tool following the standard Macintosh user interface. It features context-sensitive pop-up menus and HyperCard-like characteristics. Using single-point data entry, TurboCASE recognizes all data, symbols, and the relationships between them. A relational database captures information as it is entered so that subsequent alterations are immediately posted throughout the model. The software package also offers import/export capability, multiuser support, and desktop publishing links to the standard Macintosh "cut and paste" format as well as PageMaker and Interleaf.

C++ Utility Library
EMS Professional Shareware Libraries
4505 Buckhurst Ct.
Olney, MD 20832
(301) 924-3594

EMS Professional Shareware Libraries has begun shipping an **updated version of its C++ Utility Library**, a comprehensive collection of all ninety known public domain and shareware C++ language files for professional C++ programmers. All files are compressed with PKZIP and optimally stored onto thirteen diskettes. All products in the library (and all known commercial C++ PC products) are described and indexed in a database that accompanies the library.

C++ SPARC Development Tools
Microtec Research Inc.
2350 Mission College Blvd.
Santa Clara, CA 95054
(408) 980-1300

Microtec Research, Inc., in conjunction with Sun Microsystems, Inc., has introduced the **C++ SPARC development tools** specifically designed for embedded systems engineering. The Microtec Research tools include an optimizing C++/ANSI C cross-compiler, Assembler/Linker, a C++ source-level debugger, and a C++ symbol name inspection tool. These products were developed by Microtec Research under contract from Sun Microsystems, Inc. Microtec Research will initially offer the SPARC C++ Compiler and companion tools on SPARCstation platforms. Other hosts will follow, including the popular IBM PC.

IDB Object Database 1.1
Persistent Data Systems
75 W. Chapel Ridge Rd.
Pittsburgh, PA 15238
(412) 963-1843

Persistent Data Systems, Inc. announced **Version 1.1 of the IDB Object Database** is now available on several new platforms: Macintosh (under Systems 6 and 7), NeXT (under Version 2.0), and all HP machines running HP/UX. IDB can be configured to run stand alone or networked on any supported platform, including previously available DOS/Windows 3.0, Sun, and HP/Domain OS. With Version 1.1, IDB now supports controlled sharing of information across heterogeneous networks with any mix of supported platforms.

Send Product Announcements to Marie A. Lenzi, 225 Rector Place, No. 12N, New York, NY 10280. Fax (212)786-4032. Include company name, address, and phone number.

Databases

... As with expert systems, robotics and Materials Requirements Planning itself, it doesn't make sense to wait for object-oriented Enterprise-wide Requirements Planning systems before experimenting with the technology. Several companies offer research data bases that can handle smaller, less time-critical exercises than MRP ... Perhaps the primary reason for the lack of object-oriented applications for mainframes is the cost. Distributed, heterogeneous object-oriented applications will play havoc with the already shrinking mainframe market. While most software vendors are just now seeing financial returns on their transition from flat files to relational data bases, some major manufacturing systems vendors are still in the process of developing relational products. Rest assured, object-oriented MRP will arrive. It probably will be called something else and it will be surrounded by a mass of new management colloquialisms — but it will come ... Object-oriented planning systems will make decisions and recommendations faster than any planning team in existence today. Even the best-managed MRP sites will see astronomical improvements in inventory management and resource utilization ...

Object of Your Desire: The Future of Manufacturing Systems, Dan Rasmus, ***Manufacturing Systems****, 7/91*

... The relational model can be fully supported in an OOPL; however, the reverse is not true. Applications that need to store and share objects have needs that cannot be addressed by manipulating tables. Type definitions also change over time. Retrieved objects may be called on to perform functions added after they were stored ... Objects may also need to be constantly converted between multiple OOPLs, but neither relational nor object-oriented databases have facilities to deal with such continuous change. RDB transactions tend to be short and involve a tiny fraction of the objects in the database. Each data type has well-known operations and side effects. OODBs, on the other hand, must support a very large number of types with user-defined operations and many possible side effects ... By giving up table structure and some integrity checks, OODBs can be made thousands of times faster than RDBs. This makes them useful in applications like CAD/CAM or hypermedia, where RDBs are simply too slow.

Object Technology: The Difference between Databases, Craig Hubley, ***Computing Canada****, 8/15/91*

Languages & Tools

... One of the advantages of C++ is that it makes it easy to define new data types and give them behaviors that accommodate the real world objects they represent. With a little planning, you can create your own "language" that affords the design and coding convenience of a higher level language, while delivering the performance and control available in C ...

Client-Server and OOP, Wes Peterson, ***DBMS****, 7/91*

ToolBook 1.5 is an object-oriented programming (OOP) development system that provides serious programming tools and methods for both professional and novice programmers — all in the Windows 3.0 graphical environment. Asymetrix calls ToolBook 1.5 a software construction set for Windows and we couldn't say it better. In fact, Asymetrix has integrated ToolBook so nicely into Windows that it almost feels like a Microsoft product. ToolBook 1.5's closest competitor is HyperCard 2.1 ... Although HyperCard runs on a different platform, ToolBook wouldn't be possible without HyperCard. In many ways ToolBook is just another in a line of HyperCard supersets, a group that includes Spinnaker's +Plus and Silicon Beach's SuperCard. ToolBook 1.5 more than lives up to its touted functionality — it is faster, smarter, and better than Version 1.0 ... And with available add-on products it might cause a lot of professional developers to rethink their current development software, causing a literal renaissance of user programming with large corporations ...

ToolBook Upgrade Easier to Learn, Faster, Don Crabb, ***InfoWorld****, 8/26/91*

... Along with the routine new and existing OOP research and developments being reported at the conference, there was an underlying awareness among attendees that object-oriented technology is now mainstream. Among the 2,000+ OOPSLA/ECOOP [1990] attendees were an increased number of representatives from large organizations and corporations who came to buy products and hire consultants for significant development projects. For these commercial industry delegates, OOP has already been identified as a foundation software technology. In the past, the beginner and intermediate tutorial sessions were typically crowded while advanced tutorials had a handful of "old guard" attendees. This year was different. The advanced sessions were packed ...

... Object Logo is unique among commercial OOP languages because it is a "classless" OOP language. Object Logo makes no artificial distinction between an object and the class of which the object is an "instance." In languages like Smalltalk, classes are objects, instances of their metaclass. They are tied to the Behavior and Class classes ... In Object Logo, objects simply are themselves — a bundle of attributes and procedures, giving the object form and function. In addition, an object knows its parent or parents for purposes of inheriting attributes and behavior ... While this characteristic of being a "classless" object-oriented language may seem like an overly subtle distinction, it is profound ... If classless objects were all Object Logo had to offer, it would still be an interesting programming environment ...

... The Tigre Interface Designer is a valuable extension to Smalltalk in and of itself. When combined with the second component, Tigris, the Tigre Programming Environment empowers the developer with the capabilities to create multi-user object-oriented database applications. Tigris implements a shared, distributed, persistent object store ... The Tigre Interface Designer provides a convenient means of defining connections between interface elements and Tigris persistent database elements. In brief, the Tigre Programming environment blows the doors off the Hypercard/Oracle combo for multi-user database application development ...

Object-Oriented Programming: OOPSLA/ECOOP Reflections and New Products, Jim Salmons and Timlynn Babitsky, ***MacTech Journal****, Spring 1991*

GETTING STARTED

Perspective: investing in objects today

by Adrian J. Bowles

A recurring question heard throughout the software development industry today is, "Why objects? Why now?"

Next year, cars will be better, computers will be cheaper, some wines will be finer (and some will be vinegar), and real estate prices will go up (or down). Despite the uncertainty, this is a good time to adopt a new technology and a new way of thinking about the world.

The 1990s will see the gradual adoption of object-oriented techniques as the dominant method for systems development. The fundamental concepts of the paradigm have been in use in programming languages like Simula and Smalltalk for years. What is new is the introduction of tools to support the languages that are affordable and offer acceptable levels of performance. Also new is the use of objects in the analysis and design phases.

The expressive power of the Smalltalk environment was available years ago in the form of the Xerox and Tektronix workstations, but the price and performance were unacceptable. The widespread availability of 386/486-class machines with improved PC versions of the language now make it a viable option. The introduction of hybrid languages like Objective-C and C++ have made it easier for many shops to adapt. Borland's entry into this market with low-cost implementations of C++ and an object-oriented Pascal make it easy to get started. And, Borland's agreement with IBM to develop C++ for OS/2 provides a migration path for power PC users. Finally, the increasing cost of developers vs. environments makes it easy to justify purchase of tools that increase productivity.

Once the province of research laboratories and universities, advances in technology have brought the object-oriented paradigm to the brink of the commercial mainstream. The writers of the structured revolution (including Yourdon, Constantine, and Page-Jones) have joined the fray with object-based methods for the next generation. Industry leaders are betting heavily on the benefits of object-oriented development. Examples include:

- Object-oriented extensions to the AD/Cycle repository.
- Activity between IBM and object technology vendors like Knowledge Systems Corp. and Digitalk.
- IBM and Hewlett-Packard's response to an Open Software Foundation Request for Technology with an object database management system.
- Use of the technology by Andersen Consulting in assignments for clients like Brooklyn Union Gas.
- Companies including Compuware and Cadre rewriting existing applications in C++.

MAINSTREAM PLAYERS COMPETE WITH SPECIALISTS

Small companies with highly developed skills in niche markets are providing technology and guidance to major players. Smalltalk specialists like Knowledge Systems Corp. of Cary, NC, have supported vendors including IBM, Hewlett-Packard, and Texas Instruments. These pilot projects portend new offerings and internal efforts likely to influence the way systems are developed in the future:

- Semaphore Training reports revenues of approximately $2 million for object-oriented training. Its catalog of eighteen courses covering analysis through coding is one of the most comprehensive. Founded in 1987,

This article is adapted from an article appearing in *Advanced Software Development Research Highlights*, July 22, 1991.

Adrian J. Bowles, PhD, is Director, Advanced Software Development Research Service, New Science Associates, Southport, CT. He can be reached at (203) 259-1661.

Semaphore made an early commitment to specialize that has given them an edge in this market.

- Addison-Wesley, long a major player in technical publications, has capitalized on object-oriented technology with their training venture, Technology Exchange. A set of courses from independent authors is offered nationwide on public and in-house bases.
- General Electric has announced the formation of a training group for object-oriented technology. Its analysis and design courses are based on techniques developed by GE staff for internal development. According to Dr. Jordan Kreindler, who leads the effort, customer demand prompted the commercialization and development of their first five courses.

New Science's advice is to adopt a general object-oriented strategy now. Your strategy should address the selection of the right people, tools, project, and metrics. It should provide a timetable for adopting the technology as well as provide for the objective evaluation of its successes and failings.

PICKING THE TECHNOLOGY

By technology we mean the languages, tools, and methods (techniques) you select for your environment. The languages are the most stable of these components at the moment. You can choose from a variety of object-oriented programming languages and environments for under $500 for the PC and Mac. If your shop is already using C, you may be tempted to use C++ or Objective-C to preserve the investment you have in language training. To get the most out of the paradigm, we suggest an alternative approach.

Invest in an environment that forces your people to get out of the procedural code mind-set. Smalltalk, Actor, Eiffel, and some of the lesser-known languages are probably a better place to start. For most shops, the investment to have a few Smalltalk users will be offset by the gains in productivity. You can still use a language like C++ for the actual pilot project and production systems, but your people will adjust to the paradigm faster if they have to break their established language habits while learning the fundamentals of inheritance, polymorphism, etc.

Although there are no OOA/OOD methods as established or well supported by CASE tools as structured analysis/design and information engineering, real systems are being developed using OOA/OOD. The Coad/Yourdon approach has made it to a second edition, OOSD from IDE has CASE support from a major vendor, and GE will support its object modeling and design methods. Class-responsibilities-collaborators cards are a promising manual technique for identification of objects. Choosing any of the published methods can help your staff start to think in terms of objects, the first step to success with objects. Some CASE tools allow you to use more than one of the competing notations of object constructs. For example, ObjectMaker, from Mark V Systems, supports all the popular notations and some very obscure ones, too. Recognize that whatever technique and notation you adopt today, you will have to be prepared for change in the near future. The fundamental concepts, however, will be relatively stable.

DEVELOP A TRAINING PLAN

This is a complex problem. How many people need to be trained? The temptation in a lot of places is to send only a few people off to a class, and then expect them to teach their colleagues. This approach is not recommended. Years ago it was shown that the oldest children in a family, in general, do best in school. One explanation relevant to software training is that the oldest children have more access to the best teachers (adults, primarily their parents), while the younger children learn a lot from their older siblings, who may not be good teachers or role models. You would not send your people to a class with a teacher who only had a week of training, so you should not expect your people to be able to train each other without extensive training themselves. A few people should get a lot of training and support. If you plan to build a center of excellence to provide support internally, invest wisely to prevent the more common occurrence of a center of adequacy.

CONCLUSION

Object technology will be the dominant paradigm for software development in the 1990s. Although any specific tool or technique associated with object-oriented development is likely to undergo significant revision in the next twelve to eighteen months, the investment to learn about the technology by using it is sound. Entrance of companies like IBM, HP, and GE legitimizes a market populated by specialized players. Long-term gains appear to outweigh short-term expenses and the frustration of dealing with immature products and methods. Caution is still advisable in evaluating vendor claims; however, companies that invest today will be the first to gain competitive advantage from the technology. ≡≡

FYI

Excerpts from leading industry publications on aspects of object technology

Views

... [Borland's Philippe] Kahn caught the OOP bug in school in France during the early 1970s, drawn by the teachings of a professor he says was his most influential mentor — Niklaus Wirth, the inventor of Pascal. Watching Wirth's work evolve throughout the decade, Kahn decided in 1988 to move his own team from mainstream programming techniques to the new frontier of OOP. "It was one of the only times in the history of Borland that I forced something to happen," Kahn said ... Now, officials claim every Borland program is built with OOP, and the change has paid off in a faster stream of more reliable products ...

Kahn's Vision: Preaching the Message About OOP, Beth Freedman, ***PC Week****, 7/29/91*

... It is important to see that simply orienting yourself toward a system's view as being composed of objects does not guarantee anything about the utility, construction, organization, operation, or integrity of the result. An object-oriented view can be very useful, but it doesn't encapsulate the answers to mankind's eternal search for the meaning of life ...

Tech Talk: Parts/Elements Do Not Always Add Up with Object Orientation, Steve Gibson, ***InfoWorld****, 8/12/91*

... And if object-oriented software is really all it's cracked up to be, a company such as Borland (or, who knows, a software company no one has yet heard of) could conceivably exploit that technology in the business computer market and become the Microsoft of tomorrow. Ironically, the Apple-IBM alliance, or the Compaq-Digital consortium, could be very successful and sell lots of computers, yet still find themselves unable to control the technical evolution of the industry. Just ask IBM.

After a Decade, They're Still Computing the Future of the PC, Jonathan Weber, ***(Neptune) Asbury Park Press****, 8/12/91*

Applications

..."Smalltalk lets me concentrate on solving higher-level problems; I don't have to fight the language," [Abdul Nabi of Knowledge Systems Corp.] says. "At the end of the day I've made progress toward building the client's application instead of just tracking down pointer errors." As a result, Nabi says, he's more productive than other consultants. "I can charge two to three times what other consultants charge ... because on an average day I'm five times more productive"...

... The availability of Smalltalk implementations on a variety of platforms is a key benefit for clients, Nabi says: "Cross-platform development has become a big issue for our clients. They give developers high-power workstations, but the applications they write have to run on inexpensive PCs. Because Smalltalk is highly portable — much more portable than C — this arrangement is feasible." More portable that C? Sure. Any nontrivial C application is based on assumptions about the environment where the code will run. Memory addresses, device naming conventions, and other details vary from platform to platform. At the very least, C applications must be recompiled when ported from one platform to another. In Smalltalk, hardware and operating-system dependencies are dealt with by the run-time system. Platform dependencies are encapsulated, hidden from programmers and applications. Programs run in a "virtual Smalltalk machine," and can therefore be ported from one operating system to another without recompilation ...

... There's a downside, too: Customers can be so pleased with working prototypes that they're sometimes reluctant to pay for further development. "Software isn't automatically good just because it's written in Smalltalk," Nabi admits. "Thrown-together prototypes often contain really terrible code, tossed in for demonstration purposes. It's important to write the production version of the software using stricter rules" ...

A Competitive Edge, J.D. Hildebrand, ***UNIX Review****, 7/91*

Trends

The [IBM and Apple] venture's technological linchpin is an arcane but important new method of writing software called object-oriented programming ... For example, users will be able electronically to send each other "living" documents — fascinating amalgams of images, sounds, numbers, and text that carry with them all the software needed to make them work. The recipient will not only view the material on his or her screen but will be able as well to fiddle with the spreadsheets, edit the text, and dictate comments that the computer will record — even if his or her own machine isn't programmed to perform these tasks. Living documents could change the nature of office work as dramatically as the fax. The risks in developing such a product, however, are as vast as its promise. No one has ever designed an operating system using OOP for business computers; IBM and Apple will have to improvise. Finally, OOP is so different from traditional programming that software companies will have to relearn their business. Says Sculey: "This is something only Apple and IBM could pull off. Still, it's a big gamble, and we're betting our whole company on it" ...

The Future of the PC, Brenton R. Schlender, ***Fortune****, 8/26/91*

known something or someone becomes, the more it is put under a microscope and dissected — and it should be! This is happening more with the Object Management Group (OMG) as we speak. Members, nonmembers, users, analysts, journalists, consultants, and who know who else are developing their fractal image of what the OMG is, was, or should be. Don't get me wrong, this is great stuff and not unique to OMG, the computer industry, or any other industry.

The OMG is unique. Let's face it, trying to get computer technologists and marketers to agree to a set of rules for the future of software development before economic entrenchment dictates direction is an anomaly. Object technology was born in the basements of research and development labs and has long had this altruistic stigma. Objectphobia has been a disease in the ranks of middle and upper management in vendor and end-user organizations for years. These were the guys invited to present their views on the last day of the conference at 4:30 p.m. Armed with logic, mathematics, and analogies only a biology teacher could love, object technology suffered from the basic problem in the computer industry—too many intelligent people trying to prove it. The computer industry suffers from the tenet that theology and metaphysics are the thousand points of light and that the consumers, or people that spend money, are all test sites.

What is OMG?

OMG is a technology endorsement group, not a standards body. What we create may become de facto standards, but we are not accredited to enforce it. We don't sell software. We distribute a specification derived from commercially available technology that has been selected through an arduous, open, well-documented process in hopes that the membership and industry alike will clone it, develop it, and buy source or binary from an instantiation of it. In short, we set down rules for object technology that will make software development (to some people, life) easier, reusable, modular, and better.

Problem? Nothing evangelism won't cure. It is a fundamental problem found in any industry trying to lay down rules before money talks. It is the same problem the Knickerbocker Baseball Club of New York had in 1845 when they said it should be sixty feet from home plate to first base. They turned an obscure kids' game into an adult pastime by getting the adults to participate and showing them the way to a future financial windfall.

OMG's Role

There is widespread agreement that the OMG is trying to move an entire industry toward the development of interoperable applications. There is not, however, agreement as to how. Like any democratic forum, the OMG needs the support and commitment of its membership. Not unlike other computer trade groups, jockeying for position is a recreational sport. The vendor and user community have begun to put their trust and support behind OSF, X/Open, and UI. Object technology needs a catapulting to that level.

OMG must be recognized and endorsed by these organizations. Last, but not least, the eventual consumer, racked with consortia overload, technology panaceas, and promises of "fixed in the next revision," remains confused as to how to react.

Advice?

Again, fundamental and simple. Get committed. People who sit on the fence complain of a rash on both legs. To the vendor community: work with the other organizations you belong to and establish commitment to OMG from them. Work with your customers and help them understand the benefits of object technology. Make the commitment to support the work of these organizations, OMG included, as long as you can still prove you will make money.

To the user community: join. Get involved with OSF, UI, OMG, X/Open, and certain government agencies (e.g., NIST) to make your positions known. If you see factions forming, jump in and help them to stop, change, and agree. Don't form other factions in retaliation. Use these forums as education, positioning, and feedback and, most of all, help the computer industry to agree on some rules. That is the fundamental role of consortia and standards organizations. You, the consumers, are who decide the standards.

Increasingly, properties of information systems offer competitive advantages. Banks, insurance companies, information vendors, and consumer products companies are chosen because of their ability to quickly tailor their product to the customers' needs. Object technology will exemplify this competitive advantage by the shear nature of its design. Just as you make a cognitive shift to gain competitive advantage in your business, get involved with the computer industry to build consensus on a technology that, in parallel, will support the business and management processes of your organization.

Chris Stone is the President and CEO of the Object Management Group.

DOWN UNDER ≡≡

Object oriented in Melbourne, Australia

by William Haebich

Brian Henderson-Sellers, your usual correspondent, has asked me to contribute a summary of object orientation in my town.

It appears that the first commercial OO work began in Melbourne at the Operations Research Department of the National Mutual Life Association (managed by the author) in 1985. The OR team began experimenting, part-time, on applying Smalltalk to build an environment for the design, testing, and code generation for life insurance products. This particular project was terminated in 1989 for business, rather than technical, reasons. By that stage, a prototype had been constructed that consisted of a compiler for an insurance contract definition language, an actuarial calculation engine, and a graphical programming interface. The experience gained led, along the way, to a number of other actuarial models and some production software. These included a retirement income policy model (100,000 objects in 10 Mb) and a generalised superannuation benefits calculation model. The latter has been ported to C using a Smalltalk-to-C translator also developed in the department.

The first Smalltalk Users Group meeting took place in Melbourne in 1988. Meetings were held, more or less quarterly, at National Mutual up to December 1990. The group merged with the Object-Oriented Special Interest Group (OOSIG) at the beginning of last year due to common interests and overlapping membership. OOSIG was started by the author as an Australian Computer Society special interest group in 1989. The merged group now meets monthly and mounts beginners seminars as well. A number of more ambitious activities are being planned for next year. In particular, consideration is being given to establishing class standards. The current membership is 230 and is growing rapidly.

Over this period, OO commercial activity has spread. As an example, major developments are taking place in several branches of Australian Telecom. An Intelligent Network Testbed is being developed at Telecom Research Laboratories (TRL). This is used for prototyping future services, service creation tools, and the study of advanced network architectures. It includes an OO script language. The system is being built using C++ and Versant. Other TRL work includes the construction of a network analysis CAD tool in C++ and control software for an experimental broadband-ISDN switch in Eiffel.

Elsewhere, the SULTAN project consists of a two-year enhancement effort to a real-time embedded system using C++. The system comprises 200 classes and 130,000 lines of code. SULTAN is a communications controller, arbitrator, and expert system support for human operators.

The Network Management Systems Division is piloting a network management centre support system (fault diagnosis) using CLOS.

Several small consultancies are also active. Windows Solutions Pty. Ltd. provides training, and Simsion, Bowles, and Associates offer OO data modelling. Simsion and Bowles have recently built an OO database for a client using ONTOS.

On the academic side, most OO activity appears to be in the technical universities rather than Melbourne's equivalent of the Ivy League. Research work on Petri Net editors and general network simulators has been taking place at Deakin University since 1989. Courses in OO based on Eiffel and Smalltalk are being delivered at Monash University (Caulfield Campus), Victorian University of Technology, and Swinburne Institute.

If anyone working in the OO field plans to visit Australia, we would be pleased to hear from you. If any organisation is planning to enter the Melbourne market, we may be able to help. Contact Christine Mingins, Department of Software Development, Monash University, Caulfield Campus, 900 Dandenong Rd., Caulfield East, (03) 573-2078, or email cmingins@fcit.monash.edu. ≡≡

William Haebich is the founder of OOSIG. Currently, he is with Simsion, Bowles, and Assoc., 18th floor, 15 Collins St., Melbourne, Australia.

INDUSTRY ≡≡

The Object Management Group — a window of opportunity for everyone

by Michael Guttman & Jason Matthews

The phenomenal growth in recent months of the Object Management Group (OMG) clearly indicates that this new industry forum must be doing something right.

If the OMG can sustain its current momentum, it has the opportunity to become the single most important source for commercially viable object technology worldwide. Little of the OMG's potential public influence was in evidence a few short years ago when it was launched. After a flurry of public interest, the OMG quickly receded into to the shadows, while its members and staff quietly developed the organization's business and technical agendas.

In those almost-forgotten days of yore, the broad commercial acceptance of object technology was thought to be at least five to ten years away, even by the most religious of object zealots. After all, it was just over three years ago that the authors of this article, after an exhaustive literature search of the popular computer press, were able to dredge up only a few dozen references of any kind to the then-esoteric term "object oriented."

The goals of the OMG's founding fathers were therefore somewhat modest — a kind of object clearinghouse, manned by a few forward-thinking vendors, consultants, and users, that could act as an effective catalyst to certify commercially available technology and bring the acceptance of object technology a bit closer to the mainstream computer marketplace.

To hasten this process, the OMG fathers came up with a novel approach to organizing an industry body designed to sidestep the normally long and exhaustive process of setting formal technical standards. Formal standards groups are certainly noble; however, they sometimes give the unfortunate perception that by the time their work is finished their efforts will have been long usurped by the de facto standards set by much less altruistic, but more nimble, industry players.

Therefore, the OMG has not been organized as a formal standards group, but instead along the more flexible lines of an industry consortium. Rather than meticulously set formal standards for future systems, the OMG is, according to its bylaws, supposed to review, evaluate, and endorse already implemented systems that could deliver object solutions to commercial users in short order.

The offerings the OMG plans to endorse are those it feels are most likely to advance the early commercial adoption of object technology. Currently, this includes systems in such areas as a common object model, an object request broker (to support distributed object systems), and object databases, as well as object class libraries for windowing, graphics, memory management, communications, security, transaction management, and so on.

To give the OMG's endorsements some teeth, each member has agreed (but is not strictly required) to actively support and conform to all OMG endorsements in their own related commercial offerings. The hope is, of course, that the membership's whole-hearted support of the OMG's handpicked technologies will lead to quick adoption by users of object technology in general and the relatively painless creation of de facto industry standards.

The OMG's unusual and somewhat informal style of organization has provided it with the opportunity, unprecedented among such broad-based industry groups, to act quickly and decisively with a minimum of bureaucratic wrangling and overhead. The OMG can reach many decisions quite rapidly in small working groups without the exhaustive review processes that would be demanded of a formal standards body.

Indeed, the OMG Technical Committee (TC) has already laid out a very ambitious agenda for evaluating current object technology. One important evaluation process, for an "object request broker" (ORB) to control messaging in distributed object systems, now is almost complete. Amazingly, the process chosen by the ORB task force has already managed to yield a joint proposal from major technology submitters (HP, Sun, DEC, NCR, and HyperDesk) that could lead to general interoperability among a wide variety of implementations.

Concurrently, another important TC task force is now finishing up an "object model" that will be used as a reference model for both the ORB and all future technology submissions. Yet another group will soon be looking for submissions to flesh out the ORB with a set of "object services" (such as persistence, security, transaction processing, event handling, etc.) designed to further support interoperability among object applications.

The OMG has also wisely garnered varying degrees of interest and support from existing standards organizations such as OSF, OSI, and UNIX International, thereby uniting, rather than fracturing, the standards industry around their efforts to promote a common vision of object technology.

These moves have been part of a modest but apparently very effective marketing and public relations campaign by the OMG's management staff. In recent months, the almost monumental upswing in interest for all things "object" has also worked to create a substantial interest and attention for OMG among users, vendors, consultants, and (witness this article) the press.

As a result, the once-fledging OMG has now, all of a sudden, become the focal point of a great deal of activity much sooner than its founders probably expected. In the last six months, it has garnered the active participation of almost every major vendor and large-scale user of object technology, including Microsoft and IBM, both of whom recently joined. OMG now has well over 100 members and is adding new ones every month.

The almost "club-like" atmosphere that pervaded early OMG meetings, where everyone knew everyone else, is rapidly changing as a teeming hoard of new members and guests crowd the task force and technical committee sessions to both observe and participate. OMG activities are fast becoming a staple in the computer press, and each stage of its decision-making process is now assiduously scrutinized by industry reporters and consultants.

As long-time object technologists ourselves, we, the authors, can hardly complain about OMG's newfound, high-flying success. Our own company joined OMG last year expressly in the hope of fostering what appeared to be the only industry-wide organization that could effectively bring both vendors and users together behind object technology. By almost any measure, the OMG is well ahead of our most optimistic ambitions in this regard.

> ***Given the current torrid pace of technological evolution, the OMG is clearly one magic show not to be missed.***

However, despite our optimism we also feel that now is not the time for anyone to rest on the laurels of the OMG's current success. In fact, we think that both vendors and users should redouble their efforts within the OMG to make it as effective and broadly based as possible. Frankly, the OMG is quite possibly the only real chance to short-circuit possible years of unproductive haggling and finger-pointing in the object standards business, but it will work only if its members and natural constituents take substantial initiatives now to help make it succeed.

We also encourage the industry press and consulting community to take a more active, critical, and constructive role in publicly analyzing the activities of the OMG.

The founders, members, and staff of the OMG have given us all a real vehicle that could be used to deal effectively, early on, with the birth pangs of a very important new technology. However, the OMG's ultimate success is still far from assured. Only active participation by all members of the interested computing community can keep the OMG bandwagon rolling. All of us with a stake in object technology should get together and fight hard and long to make it work, for who knows when such an window of opportunity will come again. ≡≡

Michael Guttman is Executive Vice President and Jason Matthews is President of Genesis Development Corp., Richardson, TX.

The reality of object management

by Chris Stone

I remember like it was yesterday putting the ubiquitous frog under the microscope in my tenth grade biology class and ripping it apart to find out how it worked. This "research" consistently met with euphemisms of gross-out, a slight reversal of peristalsis that usually required a deep breath or a cold glass of water, and the anticipated shrieks from the opposite sex. Funny, but the lesson I learned in that biology class had little or nothing to do with anatomy. When you put anything under a microscope, your depiction of its unmagnified form changes. You see new things, derive new opinions, and draw one very fundamental conclusion — the more you analyze something, the more questions you raise.

Okay, so what the heck has this got to do with objects? Nothing really. It's the point of scrutiny that I am trying to make. The more ubiquitous and well

≡ Your Turn ≡

How about it? How are we doing? This is the place to send us your message. Tell us what you like and what you don't, what you'd like to see but don't — or whatever other feedback you'd like to send. Send comments to Marie A. Lenzi, 225 Rector Place, NY, NY 10280; fax to THE HOTLINE, *(212)786-4032; or, if you're ordering, send along with your order.*

Please check whichever box applies:

	Read entirely	Scanned	Found helpful	Will refer to	Didn't read
Feature	❑	❑	❑	❑	❑
Economics	❑	❑	❑	❑	❑
Choosing Object Technology	❑	❑	❑	❑	❑
MIS	❑	❑	❑	❑	❑
FYI	❑	❑	❑	❑	❑
Book Watch	❑	❑	❑	❑	❑
Product Announcements	❑	❑	❑	❑	❑
Calendar	❑	❑	❑	❑	❑

Please rate the following **1** (dislike or strongly disagree) to **10** (like or strongly agree):

Presentation of material __ Comments: _____

Accuracy of material __ Comments: _____

Overall helpfulness in your job __ Comments: _____

What else would you like to see in THE HOTLINE? _____

Any other comments? _____

_____

_____

Name, title & phone (optional): _____

hotline *on*
OBJECT-ORIENTED *technology*

❑ ***Yes,*** plug me into the latest thinking and developments in object-oriented technology. Enter me as a subscriber at the term marked below and rush me the current issue. This is a risk-free offer — I may cancel my subscription at any time and promptly receive a refund for the unused portion.

1 year (12 issues) ❑ $249

2 years (24 issues) ❑ $478 *(save $20)*

(outside US add $30 per year for air service)

Back issues @ $25 each *($27.50 foreign)*:
Vol.1, Nos. _____
Vol.2, Nos. _____

❑ Phone order
Call Subscriber Services
(212)274-0640

❑ Fax order
Fill out term and name/address information, then fax to:
(212)274-0646

❑ Bill me
❑ Check enclosed
❑ Credit card orders
❑ MasterCard
❑ Visa

Mail (and make out check, if applicable) to:
THE HOTLINE
Subscriber Services
P.O. Box 3000, Dept. HOT
Denville, NJ 07834
(foreign orders must be prepaid in US dollars drawn on a US bank)

Send me a complimentary copy of your related publication(s):

❑ THE JOURNAL OF OBJECT-ORIENTED PROGRAMMING

❑ OBJECT MAGAZINE

❑ THE C++ REPORT

Call Subscriber Services
(212)274-0640
or send order to
SIGS, 588 Broadway, #604
New York, NY 10012

❑ Send me a copy of THE INTERNATIONAL OOP DIRECTORY @ $63 ($71 foreign)
(all directory orders must be prepaid; fax or mail credit card information to SIGS, 588 Broadway, #604, NYC 10012; make check payable to OOP Directory. Foreign orders must be prepaid in US dollars drawn on US bank.)

Card# Expiration Date

Signature

Name

Company

Street/Building#

City/Province

ST/Zip/Country

Telephone

D1GC D1LC D1LC

hotline *on*
OBJECT-ORIENTED *technology*

VOLUME 3, NUMBER 1 THE MANAGER'S SOURCE FOR TRENDS, ISSUES & STRATEGIES NOVEMBER 1991

Leading the U.S. semiconductor manufacturing industry toward an object-oriented technology standard

Trendy new computer applications technology is viewed with a jaundiced eye by most semiconductor manufacturing managers who are battle-hardened veterans of the industry's economic and technological world wars.

Glenn Hollowell

They have seen highly touted computer applications claiming to be near-cure-all solutions fail miserably. Is object-oriented technology one of those snake oil remedies? Is it a good technology that is being oversold like artificial intelligence (AI)? SEMATECH has taken the position that object-oriented technology is a legitimate new thrust in computer systems technology and is leading the semiconductor industry to adopt it for computer-integrated manufacturing (CIM) applicatons.

WHY SEMATECH?

The U.S. semiconductor industry has seen its market share severely eroded and its U.S. semiconductor factory equipment manufacturers become equally overpowered. As this erosion was occurring from the 1960s through the mid-1980s, by and large the public sentiment was that any losses incurred in U.S. manufacturing industries would be made up in the services market.

This mind-set began changing as one after another of formerly U.S. dominated manufacturing businesses furnishing well paying jobs to semiskilled factory workers became noncompetitive and went out of business or were forced to move manufacturing off-shore to survive. The reality of the situation became more painfully apparent as several million manufacturing workers were displaced from these industries, and the workers' only alternative was to accept lower paying jobs in the service sector.

Many complex factors make up the formula causing this global redistribution of manufacturing prowess, but most of the people influencing business and government strategies agree that to abdicate being competitive in manufacturing technologies would doom the U.S. to eventually become a second-class power in the world economy. It is also widely held that the semiconductor industry is now, and for the foreseeable future, one of the most critical manufacturing industries to U.S. economic

continued on page 3 ...

IN THIS ISSUE

FEATURE — Leading the U.S. semiconductor manufacturing industry toward an object-oriented technology standard *Glenn Hollowell*	1	2	FROM THE EDITOR
ECONOMICS — Coping with complexity: OOPS and the economists' critique of central planning *Don Lavoie, Howard Baetjer, & William Tulloh*	6	19	BOOK WATCH
CHOOSING OBJECT TECHNOLOGY — What's the object? *Sig Rosenthal*	9	20	PRODUCT ANNOUNCEMENTS NEW PARTNERINGS & ACQUISITIONS
MIS — OOP: the MISsing link *Charles-A. Rovira*	12	23	CALENDAR
FYI — Industry publication excerpts	17	24	YOUR TURN

From the Editor



Consortia is a word we hear more and more lately. This is being reflected in industry horizontally and, to a lesser extent, vertically. One of the horizontal consortium is SEMATECH, which is made up of the U.S. government and fourteen U.S. semiconductor manufacturing companies. **Glenn Hollowell** (of SEMATECH and, primarily, Texas Instrument) discusses the why's and what's of SEMATECH and how object technology is perceived as a primary contributor to the success of the endeavor. "Leading the U.S. Semiconductor Manufacturing Industry Toward an Object-Oriented Technology Standard" also shows how technology and businesses are intimately working together to progress and to achieve success on all fronts—not just cooperating but truly exploiting the synergistics of such a state.

Ten years ago **Sig Rosenthal** taught me how to teach. An experienced and talented industry professional, he is a mathematician with a "big blue" background. Recently (over the last six months), he and I have been discussing this "object stuff." I gave him a collection of references, and we talked a good deal about objects on an ongoing basis. He recently surprised me with this article "What's the Object?" (It is my experience that this is a good portion of the professional population.) The piece is an intelligent primer for those who need the initial basics in object technology, but who have a fair grounding in the precursor concepts of structured development. It clearly shows how the concepts of object technology are evolutionary — how they are rooted in previous study and experience. It also points out some differences from and drawbacks of our previous ventures into conceptual approaches to technology. Back to basics!

MIS organizations probably have the largest software legacy software of all. Schemes and strategies for transition from and peaceful coexistence with this legacy are abundant — and mostly untried. Implementation of these strategies is not a "peace of cake," but is do-able. **Charles Rovira**, a frequent contributor to the *Hotline*, has been doing a good deal of work in the area of MIS and object technology. In "OOP: the MISsing link," Charles offers a transition implementation element (certainly not an entire strategy) and some accompanying software to work in conjunction with a primitive but prevalent CASE tool — Excelerator — and in particular the screen generator. He describes some means to translate screens to a Smalltalk/Widgets combination. Certainly no magic here, but what we should expect from transition tools—an AUTOMATED ASSIST.

At OOPSLA '91, **Howard Baetjer** sat on the Economics of Reuse panel. Howard disconcerted many attendees because he was coming from a different point of reference than most others in the audience. Howard and his colleagues, **Don Lavoie** and **William Tulloh**, are members of the Agorics project at the Center for the Study of Market Processes, Department of Economics, George Mason University, Fairfax, VA. The Agorics project focuses on the economics of software development (with a concentration on object technology) from an economist's point of view. I found that the economics of software development problem gave me a new perspective. This new perspective offered fresh insights into the primary and perennial problem of software development. The members of the Agorics project have agreed to contribute several articles to both the *Hotline* and to *Object Magazine*. We are interested in your comments on and impressions of the ideas and findings put forth by the Agorics project members. Please respond to both me and the project team.



Editor
Marie A. Lenzi

SIGS Publications, Inc. Advisory Board
Tom Atwood, *Object Design*
Grady Booch, *Rational*
George Bosworth, *Digitalk*
Brad Cox, *Information Age Consulting*
Chuck Duff, *The Whitewater Group*
Adele Goldberg, *ParcPlace Systems*
R. Jordan Kreindler, *General Electric*
Meilir Page-Jones, *Wayland Systems*
Tom Love, *Consultant*
Bertrand Meyer, *Interactive Software Engineering*
Sesha Pratap, *CenterLine Software*
P. Michael Seashols, *Versant Object Technology*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

Hotline Editorial Board
Jim Anderson, *Digitalk, Inc.*
Larry Constantine, *Consultant*
Mary E.S. Loomis, *Versant Object Technology*
Reed Phillips, *Knowledge Systems Corp.*
Trygve Reenskaug, *Taskon A/S*
Zack Urlocker, *Borland International*
Steven Weiss, *Wayland Systems*

SIGS Publications, Inc.
Richard P. Friedman, Founder & Group Publisher

Art/Production
Elisa Varian, Production Manager
Susan Culligan, Creative Director
Kristin R. Juba, Production Editor
Caren Polner, Desktop Designer

Circulation
Diane Badway, Circulation Business Manager
Kathleen Canning, Fulfillment Manager
John Schreiber, Circulation Assistant

Administration
David Chatterpaul, Bookkeeper
Suzanne W. Dinnerstein, Conference Manager
Jennifer Fischer, Assistant to the Publisher
Laura Lea Taylor, Administrative Assistant

Margherita R. Monck, General Manager

hotline on OBJECT-ORIENTED *technology*

The Hotline on Object-Oriented Technology (ISSN #1044-4319) is published monthly by SIGS Publications, Inc., 588 Broadway, NY, NY 10012, (212)274-0640. © Copyright 1991 SIGS Publications, Inc. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the U.S. Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher. Mailed First Class. Subscription rate — one year (12 issues) $249, Foreign and Canada $279. Single copy $25.

POSTMASTER: Send address changes & subscription orders to The Hotline, Subscriber Services, P.O. Box 3000, Dept HOT, Denville, NJ 07834.

Submit editorial correspondence to Marie A. Lenzi, 155 East 31st Street, #22S, New York, NY 10016. voice: (212)213-8701, fax: (212)725-0224

SIGS PUBLICATIONS

Publishers of *Hotline on Object-Oriented Technology, Journal of Object-Oriented Programming, Object Magazine, The X Journal, The C++ Report, The Smalltalk Report,* and *The International OOP Directory.*

Back issues available

Vol.2, No. 12/October '91 ≡ A modest survey of OOD approaches • *Bulman* ≡ What is a "certified" object programmer? • *Bellin* ≡ Perspective: investing in objects today • Bowles ≡ Object oriented in Melbourne, Australia • *Haebich* ≡ The Object Management Group • *Guttman & Matthews*

Vol.2, No. 11/September '91 ≡ From applications to frameworks • *Urlocker* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Getting started with object technology: efffectively planning for change • *Lorenz* ≡ Object statistics on the way • *Lenzi* ≡ On objects and bullets • *Onart*

Vol.2, No. 10/August '91 ≡ Distributed object management: improving worker productivity• *Osher* ≡ Getting the best from objects: the experience of HP • *Coleman & Hayes* ≡ Applications: EC employs object technology ... • *Stein* ≡ Capacity Planning: Fiddling while ROMs burn • *Rovira*

Vol.2, No. 9/July '91 ≡ Multimedia is everywhere! • *Weiss & Berman* ≡ Developing an object technology prototype • *Kulinek* ≡ Object-oriented capacity planning • *Rovira* ≡ How OOP has changed our developmental lifecycle • *Zeik* ≡ Modularization of the computer system • *Angenius*

Vol.2, No. 8/June '91 ≡ Domain of objects: the Object Request Broker • *Dyson* ≡ Object-based approach to user documentation • *Reeder* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Do we need object-oriented design metrics? • *Hopkins*

Vol.2, No.7/May '91 ≡ Hybrid object-oriented/functional decomposition for software engineering • *Henderson-Sellers* ≡ So, what makes object databases different? (Part 4) • *Blakey* ≡ Using the generic application to solve similar domain problems • *Gossain* ≡ Experiences using CLOS • *Hopkins* ≡ International Conference on Object-Oriented Technology, Singapore • *Bennett*

Vol.2, No.6/Apr. '91 ≡ An artist's perspective of programming • *Rovira* ≡ So, what makes object databases different? (Part 3) • *Blakey* ≡ Moving from Pascal to C++, Part 3 • *Gole* ≡ Object projects: what can go wrong • *Stewart* ≡ Reflections from LOOK-'91 • *Anders-Angenius*

Vol.2, No.5/Mar. '91 ≡ Large-scale object systems development • *Morgan & Davis* ≡ Data definition languages • *Blakey* ≡ Object disorientation • *Kerr* ≡ Moving from Pascal to C++, Part 2 • *Keller* ≡ The Object Management Group • *Barber*

Vol.2, No.4/Feb. '91 ≡ Moving from Pascal to C++ • *Sollisch* ≡ The Australian object-oriented scene • *Henderson Sellers* ≡ So, what makes object databases different? • *Blakey* ≡ Managing the evolution • *Novobilski* ≡ 1990: the year of realism • *Love* ≡ Choosing a configuration for object development • *Moody*

Vol.2, No.3/Jan. '91 Special issue: CASE and Methodology ≡ Object orientation: stop, look and listen! • *Page-Jones* ≡ The spiral model for object software development • *Wasserman & Pircher* ≡ Bootstrapping object-oriented CASE • *Wybolt* ≡ A CASE tool for object software development • *Ackroyd* ≡ Comments on "Race for the database pennant" • *Loomis*

Vol.2, No.2/Dec. '90 ≡ Waiting for OODB2 • *Durham* ≡ Object modeling • *Singh* ≡ Object-oriented COBOL — the next generation • *Adams & Lenkov* ≡ Object-oriented training in the UK • *Hopkins*

Vol.2, No.1/Nov. '90 ≡ Objects by teamwork • *Constantine* ≡ Race for the database pennant: a third-generation manifesto • *Stein* ≡ Project team descriptions *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.12/Oct. '90 ≡ Crown jewels: the value of OOT in MIS departments • *Bonar* ≡ Why is an object database faster than a relational system? • *Loomis* ≡ Visual programming object language environments • *Klausner* ≡ Hype — necessarily (or a necessary) evil? • *Adams* ≡ On Managing: Déjà Vu • *Lenzi*

Vol.1, No.11/Sep. '90 ≡ Introducing OOPS through design — it's all in the words • *Copen Goldstein* ≡ The Australian object-oriented scene • *Henderson-Sellers* ≡ Objects on tour (Review of Borland's OOP World Tour) • *Lenzi* ≡ Design principles for modular product development • *Bennett* ≡ Asset management and object-oriented technology • *Hopkins & Warboys*

Vol.1, No.10/Aug. '90 ≡ WyCASH+: An application built within an OOP environment • *McCullough & Deshler* ≡ Mathematics and methodology • *Martin* ≡ Organizing for reuse • *Goldberg & Rubin* ≡ CALS Expo–89 Conference • *Purdy*

Vol.1, No.9/July '90 ≡ Cognos and Eiffel: a cautionary tale • *Leathers* ≡ What's in an object anyway? • *Novobilski* ≡ The impact of Windows 3.0 on the OOP market • *Urlocker* ≡ Introducing reuse into the development process • *Goldberg & Rubin*

Vol.1, No.8/June '90 ≡ A taste of the USENIX C++ Conference • *Wells* ≡ Comparing relational and object database management systems part 2 — Composites, flexibility, and extensibility • *Martin* ≡ Case studies in prototyping, part 3 • *Goldberg & Rubin* ≡ Object-oriented requirements analysis • *Berard*

Vol.1, No.7/May '90 ≡ Object-oriented development: not business as usual • *Lorenz* ≡ Comparing relational and object database management systems part 1 — Single paradigm programming is the key difference • *Martin* ≡ Case studies in prototyping, part 2 • *Goldberg & Rubin* ≡ Understanding the recursive/parallel lifecycle • *Berard*

Vol.1, No.6/April '90 ≡ Lifecycle approaches • *Berard* ≡ A data manipulation language for C++ developers — a proposal • *Atwood* ≡ Case studies in prototyping, part 1 • *Goldberg & Rubin* ≡ I already learned in kindergarten all I needed to know about object-oriented design • *Ramakrishnan*

All back issues of *Hotline on Object-Oriented Technology* are available. Please call (212)274-0640 for details.

To order, see back page; for reprints of individual articles, call Michael Biggerstaff at Reprint Management Services (717) 560-2001.

Calendar

The Hotline Calendar presents conferences and meetings that focus exclusively on object-oriented technology. To have a meeting or conference listed, please send the dates, conference name and location, sponsor(s) and contact name and telephone number to the Editor: Marie A. Lenzi, 133 East 31st Street, #22S, New York, NY 10016, fax (212) 725-0224.

November 5–7 **Database World** Chicago, IL Sponsor: DCI Contact: Customer Service, (508) 470-3880	**November 18–22** **C++ at Work** Santa Clara, CA Sponsor: *The C++ Report* & Wang Institute of Boston University Contact: Suzanne W. Dinnerstein, (212) 274-0640	**February 3–5, 1992** **LOOK '92** Copenhagen, Denmark Sponsor: Niveau, *Object Magazine*, & *The Journal of Object-Oriented Programming* Contact: Miles Silverman (212) 274-0640	**February 5–7, 1992** **OOP '92** Munich, Germany Sponsor: *The Journal of Object-Oriented Programming & Object Magazine* Contact: Miles Silverman (212) 274-0640	**February 11–21, 1992** **C++ Across America** SF, LA, Dallas, Chic., Bos., NYC, Wash. D.C. Presented by: *The C++ Report* Sponsored by: Microsoft Contact: (212)274-9135

New Partnerings and Acquisitions, November 1991

ParcPlace Systems has announced its support of **Information Builders'** Enterprise Data Access/SQL product family. ParcPlace Systems intends to extend its suite of Objectworks/Smalltalk portable objects (object-oriented class libraries) to provide a common interface for applications that use IBI's EDA/SQL product. They have signed a strategic marketing partner agreement wherein ParcPlace will use EDA/SQL as its primary database connectivity solution, while IBI has agreed to support Objectworks/Smalltalk in its marketing efforts.

Tigre Object Systems, Inc. is now bundling the Tigre Programming Environment with **ParcPlace Systems'** Objectworks\Smalltalk object-oriented language for a reduced package price through 1991. With the bundled price, customers save $1,900 as well as time: Tigre comes preinstalled and ready to run.

Digitalk, Inc. has become a member of the **IBM** International Alliance for AD/Cycle. The move underscores IBM's commitment to Smalltalk/V within Ad/Cycle and is seen as providing key tools for delivering Common User Access (CUA '91)-compliant applications within AD/Cycle. In a related action, Digitalk has confirmed that IBM has signed a letter of intent to market Digitalk's Smalltalk/V products to IBM customers worldwide. IBM and Digitalk have worked closely together to focus Digitalk's enhancement of Smalltalk/V PM.

Object Design, Inc. has licensed **PRC** of McLean, VA to exclusively market ObjectStore, Object Design's object-oriented database management system, to U.S. government agencies and contractors for those agencies. PRC offers comprehensive imaging integration services to both the federal government and commercial marketplace.

IBM Corp. and **Sapiens International Corp.** are teaming up to jointly market SAPIENS, an application development and maintenance product that uses rule-based, object-oriented technology to build and execute data processing applications on IBM mainframes.

Gene Wang, Vice President and General Manager of the Languages Business Unit of **Borland International, Inc.** has been named to the Board of Directors of the **Object Management Group** (OMG). Before joining Borland, Wang co-founded Gold Hill Computers, serving as Vice President of Marketing. Prior to this, he was a Project Leader in research and development for Wang Laboratories (no relation).

International Business Machines Corporation has signed a licensing agreement with **Servio Corporation** for the GemStone object database management system. IBM plans to use GemStone in a manufacturing automation application. Financial details of the agreement were not disclosed.

In a stock swap worth over $12 million, **Symantec Corp.** has acquired languages developer **Zortech Inc.** in a bid to boost the PC utilities company's presence in the burgeoning object-oriented languages market. Symantec will use the Zortech line to promote a cross-platform solutions for object-oriented developers.

Oberon Software Inc. of Cambridge, MA and **Toyo Information Systems** (TIS) of Osaka, Japan have signed a distribution and license agreement for Oberon's SynchroWorks object-oriented software environment. Under the terms of the agreement, TIS will pay Oberon $3 million for the rights to distribute SynchroWorks in the Pacific Rim area (excluding Australia and New Zealand) and will pay Oberon royalties for the use of SynchroWorks in systems integration and software package development projects.

SCO UNIX SYSTEM V /386, from **The Santa Cruz Operation, Inc.**, has received POSIX certification from the **National Institute of Standards and Technology**. This is the first hardware-independent operating system to receive this certification. POSIX certification assures government and other users that software developed on SCO UNIX System V for 386 and 486 machines will be source compatible with other POSIX-certified operating environments. SCO also announced its endorsement of the new **Open Software Foundation**'s Distributed Computing Environment technology.

AICorp, Inc. has expanded its local partner network to Portugal and Spain through an agreement with **Grupo Formula**. Under the terms of the agreement, Grupo Formula will distribute AICorp's KBMS, the Knowledge Base Management System. The agreement is an expansion of AICorp's relationship with Grupo Formula's parent company, which distributes AICorp's products in Italy.

Software for Columbus, the **European Space Agency**'s Space Station, will be developed using Software BackPlane, **Atherton Technology**'s CASE software. BackPlane is the basis for the Columbus Software Development Environment, which will be used by software developers at the more than thirty companies working on the Columbus project.

... continued from page 1

interests because it is the beginning of the technology food chain for dozens of technology-driven fields. Early semiconductor leadership is directly responsible for U.S. leadership in computers, military electronics, much of the modern medical equipment advances, and many other technology-oriented enterprises. Most semiconductor industry pundits say the U.S. semiconductor industry did not lose its edge because of a lack of product creativity, but from lethargy in manufacturing, particularly in productization — the ability to bring products from design to full production.

As a result of the recognition of these conditions, in 1987 the U.S. Congress authorized the creation of SEMATECH — a consortium made up of the U.S. government and fourteen U.S. semiconductor manufacturing companies — to develop and share generic precompetitive technology to increase competitiveness. The commercial participants in the consortium include companies whose primary business is the manufacture and sale of semiconductors and those who manufacture semiconductors generally for use in their finished product. The members who are primarily semiconductor suppliers are Advanced Micro Devices, Harris Semiconductor, Intel, LSI Logic, Micron Technology, Motorola, National Semiconductor, and Texas Instruments. The members that are usually known more for their end products than their semiconductor manufacturing are AT&T, Digital, Hewlett-Packard, IBM, NCR, and Rockwell.

WHY STANDARDIZE SOFTWARE TECHNOLOGY?

Participating SEMATECH members determined early that CIM had to be a primary contributor to a multipronged strategy to improve manufacturing productivity and revitalize U.S. semiconductor equipment suppliers. It is recognized that the use of computer systems in almost every facet of semiconductor manufacturing is absolutely necessary to remain competitive. All of the semiconductor manufacturers and semiconductor manufacturing equipment suppliers certainly understand that requirement. But, the emerging era is one in which standalone computerization, islands of automation, or even interfaced systems are not enough. To remain competitive, a semiconductor manufacturer will have to incorporate CIM systems that are highly integrated. Machines will have to be responsive to work cells, work cells must be well-coordinated appendages of the factory controller, and factory controllers will need to be attuned to the needs of enterprise-wide systems and customers' systems. Yes, the customers' systems! Electronic data interchange (EDI) technology is rapidly moving to the point where customers will expect, as a condition of doing business, that their systems will have an electronic window into their suppliers' factories where they can see the progress and problems of products that are being manufactured for them.

With requirements such as these, it was equally clear that technology as usual in the application of CIM would not provide significant competitive advantages over foreign competition because of the litany of problems that current technology has failed to resolve. The most serious of the problems being faced includes the lack of interoperability, data inconsistency and redundancy, complex monolithic applications that are difficult to change and maintain, difficulty of applications extensibility, and lack of models for software engineering, maintenance, and support. As important as the current quality movement is to U.S. industry, it is only a precursor to a much more difficult requirement of maintaining an absolute consistency in quality excellence while being quick to respond to changing market trends and customer needs.

Consequently, SEMATECH instituted its Strategic Development Program and began an extensive process of evaluating, specifying, and adapting various computer-based methodologies and technologies to determine which would provide the best potential to achieve step function gains in improving development time, incorporating greater intelligence, lowering human resource consumption, and reducing maintenance requirements in CIM applications. Pure CIM applications were not the only considerations for the program because manufacturing is only a single part of the broader community of enterprise systems within which it must cooperate, integrate, and compete for resources. As greater numbers of systems are developed and deployed to manage increasingly complex businesses, legacy systems, both manufacturing and traditional information systems, are consuming larger and larger portions of available resources because of the aforementioned problems of maintaining the current applications technology. It is generally conceded that maintenance and support requirements for legacy systems has grown to consume 80% of the human resources in today's systems organizations. All the while, demand for the development of new applications to manage greater complexity is climbing exponentially.

ADOPTION OF OBJECT-ORIENTED TECHNOLOGY

The most far-reaching decision resulting from the SEMATECH Strategic Development Program has been the choice of object-oriented technology as the foundation technology for its CIM architecture.

In 1989, when SEMATECH's Strategic Development Program began making development technology decisions, most of what was heard about object-oriented technology was largely increased productivity claims that could not be substantiated in actual applications accomplishments. Yet, several prominent U.S. computer companies, who were also SEMATECH member com-

panies, encouraged SEMATECH's adoption of the technology. Also, another member company had started developing an unconventional next generation semiconductor manufacturing facility under contract to the U.S. government about one year earlier. They had chosen object-oriented technology for the CIM aspects of that project and had already shown potential for vast improvements over conventional software technology. These member companies' advocacy of the technology along with the claims of large software development and maintenance efficiencies influenced a tentative endorsement by SEMATECH. Since then, confidence in the technology has come with successful usage and has grown to the point that the object-oriented criteria is one of the determining factors when making selection decisions on most system methodologies or products.

Object-oriented analysis and design

Many organizations that begin dabbling in object-oriented technology start by experimenting with object-oriented languages such as C++ and Smalltalk. They start using the languages before they understand that the fundamental principle of object-oriented technology is that real success depends on objects being the focus from the very start of the development life cycle — in analysis and design. When an organization tries to implement object-oriented programming languages without the necessary changes in analysis and design principles, the typical results range from total incomprehension and turn-off to only using the language in a convoluted non-object-oriented manner. Fortunately, SEMATECH did not fall into that trap. In fact, SEMATECH didn't even concern itself at that stage with the question of which object-oriented language to adopt. Instead, SEMATECH initiated a comprehensive study and evaluation into analysis and design methodologies and supporting tools. One of the things most conclusively shown in the study was that object-oriented methodologies and tools are at an extremely early stage in the technology life cycle, and the marketplace of object-oriented ideas and products is expanding rapidly. Many of the established methodologists and tool suppliers have only recently decided that object-oriented technology has the potential equivalent impact in the '90s that was experienced by the "structured" analysis, design, and programming movement of the '70s and '80s. They are only now developing capabilities to accommodate object-oriented techniques.

In the very early stages of its CIM Architecture Program, SEMATECH established a set of guidelines for a requirements analysis phase as well as separate external and internal design phases. In this context, SEMATECH's definition of an external design methodology is one that maintains a "level of independence from the target programming language," while internal design allows for all of the implementation-dependent variables. This is not as easy as it might sound because many of the methodologies that claim language independence are quite heavily biased by specific language techniques.

SEMATECH has gone to great lengths to differentiate between the external and internal design phases because many of its development plans call for projects to have their requirements analysis and external design phases performed internally or by groups of member company people that may be made up of competitors, but for the applications development and implementation be out-sourced via competitive bids. Plus, external design is considered "precompetitive" to the point that member company people from competing companies, who might ultimately be bidders for the development contract, can work together on an external design specification.

Of the fifteen methodologies that were studied for a requirements analysis methodology and of the eighteen candidates for a design methodology, the Shlaer-Mellor methodology was chosen for requirements analysis, and the Booch '90 methodology (as opposed to Booch '86, which is not object-oriented) was chosen for the design methodology. Some of the studied candidates were eliminated because they did not meet the "show stopper" requirement of "support for an object-oriented approach," some were eliminated because they were semantic abstractions with no commercial tool support and open to much interpretation, and the finalists were judged on desirable features. When two different methodologies were chosen for requirements analysis and design, it was recognized there would be a certain amount of translation required to be performed between the phases. These two methodologies are still the "official" choices, but experience has resulted in a strong movement toward the use of a single methodology made up of a combination of several methodologies, which are discussed later.

The initial requirements analysis specification for semiconductor manufacturing applications took the form of member company contributors building manufacturing scenarios that were taken by SEMATECH's staff and developed into the object-oriented requirements and external design models. This may seem a little strange because normal requirements analysis is typically done by analysts visiting the factory to interview key people and perform first-hand observation. This was prevented from happening because SEMATECH is a consortium of competing companies, and these companies are very reluctant to open their plants to the level of scrutiny necessary to perform a requirements analysis. Even so, the member companies wanted SEMATECH to develop a next-generation demonstration system that could be used as a model for the industry. So, scenario building was devised as a compromise to provide the information without opening the factories to outside observers.

CommLib Level 2 Version 3.2
Greenleaf Software, Inc.
16479 Dallas Pkwy.
Dallas, TX 75248
(214) 248-2561
(800) 523-9830

Version 3.2 of Greenleaf CommLib Level 2 has been released. In its last release of CommLib, Greenleaf incorporated a new layered architecture that provides transparent access to low-level drivers using a technique similar to C++ virtual functions. Version 3.2 adds seven new drivers to CommLib, bringing the total up to twelve. Version 3.2 also provides improvement to Kermit by adding the ability to selectively use data compressions and support for eight-bit prefixing and text mode transfers. Greenleaf also added drivers for two popular DOS extenders — the Phar Lap 286/DOS-Extender and Rational Systems' DOS/16M package.

zApp
Inmark Development Corp.
2065 Landings Dr.
Mountain View, CA 94043
(415) 691-9000

Inmark Development Corporation has announced the release of **zApp**, a robust applications framework designed to help programmers quickly and easily develop Microsoft Windows applications. zApp was developed to provide the functionality needed to create even the most complex windows applications by encapsulating the Windows API into easy-to-use C++ objects. zApp can also be used with existing C code, providing object-oriented programming benefits during the enhancement of current applications.

Teamwork/OOD
Cadre Technologies, Inc.
222 Richmond St.
Providence, RI 02903
(401) 351-CASE

Cadre Technologies has introduced **Team*work*/OOD** for object-oriented design. Team*work*/OOD consists of a graphical editor, a C++ code frame generator, a C++ code capture utility, and an operational interface to the popular Saber-C++ programming environment from Saber Software Inc. It is fully integrated with Team*work* for OOA and with Team*work* facilities that provide an enhanced common data dictionary, automatic document production, model management, and a programmable interface.

Also announced was the availability through Cadre of extensive methods training and tool support for Project Technology Inc.'s Recursive Design, a language-independent design method.

Baby Driver Printer Interface Library 2.0
Autumn Hill Software Inc.
1145 Ithaca Dr.
Boulder, CO 80303
(303) 494-8865

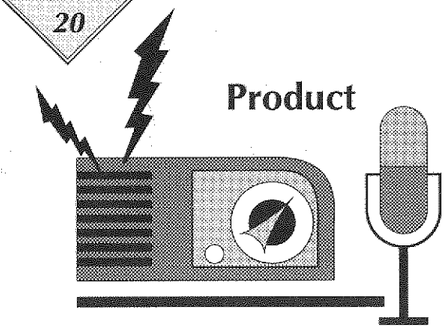
Autumn Hill Software has begun shipping **Release 2.0** of its **Baby Driver Printer Interface Library**, a C and C++ developer's tool for DOS and 386 protected mode environments. Release 2 provides support for over 300 printers of all kinds and implements that support with a table-based, smart driver that can process printer command constructs containing arithmetic and logical expressions, conditional logic, loops, and array variables. This architecture is designed to deal with the most arcane printer command interface without resorting to special coding or drivers. The Baby Driver 2 Kernel provides both graphic and text capabilities, but is especially strong at rendering graphics images. The product supports unlimited color palettes and incorporates a variety of dithering technologies, which allows an application to extract the greatest attainable result on a given printer. A separate Presentation Graphics option is available that adds a complete graphics primitive library designed to drive any printer at its maximum resolution.

Liana
Base Technology
1543 Pine St.
Boulder, CO 80302
(303) 440-4558

Base Technology has released **Liana**, an application development tool for Microsoft Windows. Liana is a C-like object-oriented programming language and class library designed specifically to simplify the development of Windows applications. The syntax of Liana is similar to C++ but simplified and extended so that it has the friendliness associated with languages such as BASIC but with the power and conciseness of C and C++. The high-level class library hides the tedious details of the Windows API and provides convenient access to features such as DDE, MDI, DLLs, and most other capabilities of Windows 3. Simplified string operations, automatic memory management, dynamic arrays, and the high-level class library are a few of Liana's features.

AppMaker
Bowers Development Corp.
PO Box 9
Lincoln Center, MA 01773
(508) 369-8175

Bowers Development Corp. is offering a free update of its interface building product, **AppMaker**, which will provide support for THINK C 5.0 and THINK Pascal 4.0 users. AppMaker cuts development time by automatically programming the Macintosh user interface for an application. The user automatically generates ready-to-run source code for selected modules or for the entire application. AppMaker allows experienced programmers to generate the code for the user interface in a fraction of the time they would have spent writing the code by "hand." And, beginning programmers can study AppMaker's generated code to learn new programming languages, especially OOP languages. MS-DOS programmers can use AppMaker to generate the Macintosh user interface code and then write the code for the rest of their application.



Product Announcements

Product Announcements is a service to the readers of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY; it is neither a recommendation nor an endorsement of any product discussed.

C++/SoftBench

Hewlett-Packard
19319 Pruneridge Ave.
Cupertino, CA 95014
(800) 752-0900

Hewlett-Packard's **C++/SoftBench** now runs on Sun SPARCstation computers. C++/SoftBench provides an integrated, extensible environment for program development and tool integration in C++ and C languages. Users can share development tools and data across Sun and HP computer systems without expertise in the detailed use of each system. New object classes can be constructed visually; the underlying code templates are generated transparently to the user, however, the programmer can check source code easily for consistency with design rules.

SPARCworks Professional Series

SunPro
2550 Garcia Ave.
Mountain View, CA 94043
(415) 336-3566

SunPro, a new Sun Microsystems, Inc. business, has released new versions of its SPARCworks and SPARCompiler products. The products, known as the **SPARCworks Professional Series**, offer significant enhancements for C++, ANSI C, Fortran, and Pascal developers. They are the industry's first compilers and productivity tools developed for Solaris 2.0, the recently introduced UNIX System V Release 4 (SVR4)-based operating environment from SunSoft. The new features in SPARCworks simplify program development by supplying tools for browsing and graphically displaying the program structure, including C++ class hierarchies, debugging optimized code and shared libraries, collecting and analyzing application and system performance data, comparing and merging multiple versions of files, viewing and controlling system build procedures, and launching and managing individual session tasks.

Actor 4.0/ ObjectGraphics

The Whitewater Group
1800 Ridge Ave.
Evanston, IL 60201
(708) 328-3800

Whitewater has announced **Release 4.0** of its **Actor** object-oriented programming platform. Actor is available in two versions: Actor 4.0 and Actor 4.0 Professional. Actor 4.0 is perfect for users who want to learn Windows development and object-oriented programming. For developers who need SQL access, Actor 4.0 Professional adds SQL libraries for dBASE, Excel, Paradox, and ASCII plus the ObjectGraphics library and the Whitewater Resource Toolkit. For an additional fee users can purchase DLLs for SQL Server, DB2, Oracle, and OS/2 Extended Edition Database. An Actor class library provides uniform access across all databases.

Whitewater has also announced the newest version of **ObjectGraphics**, which extends the power of **Turbo Pascal For Windows**. This version is designed to be the natural extension of the ObjectWindows class library found in Turbo Pascal for Windows. ObjectGraphics is the first in a series of increasingly higher-level class libraries that will use ObjectWindows as a foundation and make platform-independent GUI applications possible.

Rational Rose

Rational
3320 Scott Blvd.
Santa Clara, CA 95054
(408) 496-3692

Rational has announced **Rational Rose**, a graphical object-oriented analysis and design tool that supports software engineers developing object-oriented applications. A client/server application, Rational Rose enables language-independent analysis and design and is available for the IBM RISC System/6000 and the Sun SPARCstation UNIX workstations. Rose can also be accessed from any X Window System-compliant display running Motif or OpenWindows.

Windows 3.0 version of Procyon Common Lisp

ExperTelligence, Inc.
5638 Hollister Ave.
Goleta, CA 93117
(805) 967-1797

ExperTelligence, Inc. has released the **Windows 3.0 version of Procyon Common Lisp**, a full Common Lisp object system available for 386 and compatible microcomputers. The product, which runs under MS-DOS in the Windows 3.0 graphical environment, was developed by Procyon Research Ltd. of Cambridge, England. In addition to full Common Lisp, Procyon offers a full CLOS object system that includes multiple inheritance, a range of inheritance and method combination mechanisms, Before/After/Around method types, multiple method dispatch, and meta object protocol. A graphical layer allows the porting of standard interface objects from Macintosh to Windows and, soon, to OS/2 Presentation Manager. The development environment is totally interactive and integrated with the windows system.

Send Product Announcements to Marie A. Lenzi, 155 East 31st, #22S, New York, NY 10016.
Fax (212)725-0224. Include company name, address, and phone number.

Object-oriented CASE and beyond

Along with the object-oriented methodologies, the study included an appropriate examination of computer-aided software engineering (CASE) tools to support the selected methodologies. ObjectMaker (formerly named ADAGEN) from Mark V Systems was selected and has been in use at SEMATECH for about a year. It should be noted that none of the candidates scored high on having all of the functionalities that were truly needed, but two of ObjectMaker's strongest competitive factors were the breadth of methodologies it supported, both object-oriented and structured, and its ability to easily encompass new methodologies or be modified to use combinations of several different methodologies. This flexibility is accomplished by the methodologies being implemented through a rule language rather than being imbedded in the tool's software. As it turned out, this functionality was put to more use than originally expected because the methodology that evolved for actual usage was a combination of the Booch '90 methodology, Harel state charts, some Colbert notations, and a few SEMATECH-specific features.

SEMATECH's follow up to the CASE and methodology study is an extensive evaluation of object-oriented framework tools. A framework is a tool that provides a shell or environment under which various incompatible software development tools, such as CASE, configuration management, project management, documentation, etc., can be integrated to work together through various data translation and repository services. This evaluation is targeted to be completed so that SEMATECH can pilot a framework on at least one internal project beginning in the first half of '92. The advances in object-oriented technology, plus the high level of concern in improving software quality through organizations such as the Software Engineering Institute (SEI), the European Strategic Programme for Research of Information Technology (ESPRIT), and Software Technology for Adaptable, Reliable Systems (STARS) projects, have become the catalysts that are feeding a frenzy of framework activity. Frameworks technology is still very immature, the field is still wide open for new capabilities, and many new advances will be introduced in the next couple of years.

Similarly, SEMATECH is heavily involved in work to improve the software development life cycle of advanced semiconductor CIM applications. Work to date centers around the specification of an explicit life cycle model with clear definitions of its various phases and components. This work is continuing with pilot applications both internal to SEMATECH and in member companies.

Object-oriented execution environment

One of the earliest and highest priority action items of SEMATECH's CIM Architecture Program was to define an execution environment that would provide highly portable applications across the industry. The vision is one where semiconductor manufacturers can build or buy from a semiconductor equipment supplier a CIM application that has a high level of portability and interoperability across a wide range of heterogeneous hardware architectures, configurations, and operating systems. This was originally defined to be accomplished though the use of a SEMATECH-specified application programming interface (API) and a set of standards. SEMATECH developed the system requirements specifications for such an execution environment, but discontinued this development program when a combination of broad-based industry organizations appeared to be close to resolving this issue in an acceptable manner and time frame. Primarily, SEMATECH is looking to the Object Management Group (OMG) and Open Systems Foundation (OSF) to provide these capabilities. It is felt that the OMG and OSF combination has a high probability of delivering much of the capabilities defined in SEMATECH's execution environment requirements specifications because of OSF's declared intention to implement OMG's Object Request Broker (ORB) and Object Model standards into the OSF's distributed environment. When SEMATECH made this systems architecture policy decision, it became a member of OMG and OSF and has adopted a proactive role in both organizations.

SUMMARY

In addition to the commonly claimed benefits of the object-oriented technology, SEMATECH has gained other advantages from adopting it as a standard. The technology has been conducive to providing a common thread across the entirety of its system life cycle. Plus, object-oriented technology is especially applicable to CIM systems because physical manufacturing objects from the user's world and application objects from the abstract software world can be closely aligned.

The decision to adopt object-oriented technology has not been without its difficulties. It would have been much more expedient to embrace the time-tested structured systems methodologies, but if SEMATECH is to provide the vision for step-function improvements in manufacturing technology, it must take enlightened risks. These decisions and pioneering activities in object-oriented technology already appear to be having a substantial positive impact on SEMATECH's member companies' willingness to put object technology to real use in CIM applications.

Studies, evaluations, specifications, and models are all appropriate in their place, but the results-oriented management in the semiconductor industry will require a successful demonstration of real benefits for SEMATECH's CIM Architecture Program to achieve industry approval. A difficult, but achievable, task. ≡≡

Glenn Hollowell is a Senior Member, Technical Staff of Texas Instruments, one of the fourteen member companies of the SEMATECH consortium. Glenn also serves SEMATECH as a member company consultant in areas related to advanced computer-integrated manufacturing (CIM) applications development and execution environments. He has neen involved in most of SEMATECH's evolution to its current firm committment to object-oriented technology. Glenn can be reached on Internet at 4333722@mcimail.com.

ECONOMICS ≡ ≡

Coping with complexity: OOPS and the eonomists' critique of central planning

by Don Lavoie, Howard Baetjer, & William Tulloh

The concerns of a business manager debating the use of object-oriented technologies may seem far removed from recent political and economic convulsions in the Soviet Union, but we economists find some significant parallels.

The colossal failure of Soviet-type economies may be attributable to their violation of the principles of object-orientedness.

The classic critiques by such economists as Ludwig von Mises[1] and Friedrich A. Hayek[2] of centralized economic planning emphasize the role of the division of labor, property rights, and markets in enabling economies to cope with complexity. Such institutions can be seen as facilitating the use of knowledge in society in much the same way that object-orientedness facilitates the use of knowledge in software. The benefits of reform in former command economies and the benefits of adopting an object-oriented approach to software development can be seen as stemming from a common root. Programming practice, like economies, have evolved techniques to help coordinate complex systems. We are struck by the similarities between these programming techniques and evolved economic institutions. By examining these similarities, as well as remaining differences, we can perhaps gain a better perspective on how object-oriented technologies help software development teams cope with the growing complexity of their task, including the need to respond quickly to constantly changing demands placed upon these systems.

Don Lavoie is Associate Professor of Economics at George Mason University. He is a leading critic of centrally planned economies and co-director (with Mark S. Miller) of the Agorics Project, a research effort sponsored by the Center for the Study of Market Processes and aimed at investigating issues that overlap between computer science and economics. Howard Baetjer and William Tulloh are graduate students in the Economics Department at George Mason University and members of the Agorics Project.

MONOLITHIC PROGRAMS AND ECONOMIC SELF-SUFFICIENCY

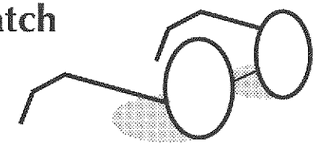
In the early days of computers, when power and memory were scarce, only very simple programs could be written. These were simple enough, in fact, for a single programmer to conceive, plan, and construct in a simple, monolithic fashion. Every detail of function and data was immediately under the command of the programmer. This state of affairs may be likened to production by Robinson Crusoe, whose activities must be kept simple enough so that they might all be managed and carried out by a single person. Robinson Crusoe, for instance, must make his own bow, harden the tips of his own arrows, and do all his own hunting, skinning, and cooking. Crusoe's standard of living is severely limited by the scope of the production tasks he can undertake. Similarly, this monolithic approach to programming, in which the whole process is controlled by a single mind, quickly reaches its limits in the scope and complexity of the programs that can be created.

FUNCTIONAL DECOMPOSITION AND THE DIVISION OF LABOR

With the continuing decrease in the cost of memory and processing time, programmers' ambitions and users' demands quickly outstrip the ability to construct programs in such a simple, monolithic fashion. Programmers needed to undertake more complex tasks with a variety of functions and interrelationships. But they found themselves unable to manage these systems in the old, monolithic fashion. The systems they tried to devise were incomprehensible with this approach, simply exceeding the mind's capacity to make sense of so much detail all at once. In response, programmers began to divide the tasks into more manageable parts, into subroutines.

This innovation can be seen as the recognition by programmers of the principle of the division of labor. The division of tasks within a program allows the programmer to keep effectively in mind greater levels of complexity. But, along with this decomposition of the problem into tasks, comes the need for coordination among the various tasks.

Book Watch



The "Book Watch" column does not contain book reviews. These listings are abstacted from press releases provided by the publishers, and no endorsement is implied. Please send announcements to the Editor: Marie Lenzi, 225 Rector Place, Suite 12N, New York, NY 10280, fax (212) 786-4032.

The C++ Programming Language, Second Edition

Bjarne Stroustrup; Addison-Wesley, Reading, MA $34.50

This new edition has been completely rewritten and greatly expanded to help serious programmers use C++ for nontrivial projects. Coverage of all C++ features, including exception handling, templates (parameterized types), and multiple inheritance, is provided.

C++ Primer, Second Edition

Stanley B. Lippman; Addison-Wesley, Reading, MA $32.25

This new edition covers C++ as reflected in its latest release, Release 3.0, and the ANSI base document, *The Annotated C++ Reference Manual*. As a leading member of the team working on the implementation of Release 3.0, Lippman provides invaluable insights into the new language features.

Advanced C++ Programming Styles and Idioms

James O. Coplien; Addison-Wesley, Reading, MA $33.50

Assuming a background in C++ basics, this book helps programmers to develop their programming expertise by giving a feel for the styles and idioms of the language. The approach is organized around the abstractions that C++ supports: abstract data types, combining types in inheritance structures, object-oriented programming, and multiple inheritance.

... FYI, continued from page 18

Languages & Tools

... Different languages provide a variety of tools positioned along a spectrum for aiding software development. At one end of the spectrum are languages like C++, in which the only tools are external development environments. In the middle are languages like Smalltalk that provide residential development environments having access to every part of the language and its implementation. At the other end of the spectrum are languages like CLOS that provide linguistic mechanisms to support development ...

LISP, John K. Foderaro, ***Communications of the ACM****, 9/91*

... [V]endors must demonstrate that object-orientation is a better solution than a relational database hybrid that contains object-oriented extensions or allows users to define data types called BLOBS — binary large objects. The later efforts fall short of true object orientation, said IDC analyst Steve McClure. "BLOBs are basically buckets of bits — the database still has no idea of what's in the bucket and still has to locate them by a unique reference — that's not object orientation," he said ...

Object-oriented niche, Barbara Darrow, ***Computer Reseller News****, 9/2/91*

... Visual Basic comes agonizingly close to being a real object-oriented language, but misses the mark by omitting vital object-oriented programming capabilities. You can bind methods to the existing user-interface objects, but there's no way to create new classes. If you want ten red buttons that beep when they're pushed, you'll have to color each one individually and group them to share the beeping code — much clumsier than creating a "red beeping button" class. Instead, I'd recommend Borland's Turbo Pascal for Windows if you want real OOP functionality ...

Developers toolbox: BASIC visualization, Bret Glass, ***PC World****, 9/91*

... The object oriented model will optimally team the human brain and an artificial intelligence engine as a human interface because it uses objects for abstraction and has the advantage of visualization ...

Artificial intelligence can reduce product cost by optimizing design decisions, John Izuchukwu, ***Industrial Engineering****, 8/91*

... From their demonstrations, I have no doubt that the proprietary Windows development offerings such as Realizer, GFA Basic, VZ Programmer, and, to a lesser degree, Actor and the two Smalltalks, can do great things in the hands of their developers and those who invest heavily in learning and understanding the programs' various strengths and weaknesses. However, I didn't find myself as instantly hooked by any of them as I was by Visual Basic and QuickC for Windows ...

Windows developers finally have the right tools for the job, Steve Gibson, ***InfoWorld****, 9/16/91*

OOP at Work

Biosphere 2 ... is an experiment in closed system ecology. It is a steel and glass structure about the size of three football fields and has a volume of more than three million cubic feet. The purpose of the project is to demonstrate the viability of materially closed ecosystems — a sort of bioregenerative life support system—where water, air, and food are recycled. Later this year eight people...will be sealed inside Biosphere 2 for two years with only power and information being exchanged with the outside. A major part of this project is an expert system-based environmental control and monitoring system called the "Nerve System" ... To fulfill this requirement, a Nerve System consisting of a broad-band network, several HP9000 work stations, and a control and monitoring hierarchy of expert systems was developed using G2, a Lisp-based, real-time, object-oriented expert system development environment from Gensym Corporation. After several available traditional control systems were reviewed, G2 was chosen for the project because of its ease of use and its high degree of integration between rules, object, and graphical displays. Also, G2 rules are interruptible, thus allowing the developer to control data acquisition and response times. Controlling the heating, ventilation and air-conditioning ... systems of Biosphere 2 is a typical real-time control problem. In the past, Lisp would not have been considered a candidate to solve this problem due to response time limitations imposed by garbage collection. G2, however is carefully designed to eliminate the garbage collection of Lisp, thus reducing the chance of a delayed reaction at a critical moment ...

Biosphere 2 nerve system, Rocky L. Stewart, ***Communications of the ACM****, 9/91*

Industry News

... The [IntelliCorp] acquisition makes sense because it gives KnowledgeWare a chance to integrate its software analysis, design, reverse-engineering and code-generation tools with the object-oriented programming (OOP) capabilities of IntelliCorp's expert system tools, said Natasha Krol, an application development strategies analyst with Meta Group..."It was only a matter of time before CASE tool developers combined their technology with object-oriented programming technology," Krol said. But it will be "extremely difficult to integrate" KnowledgeWare's current product line, which is based on traditional procedural programming methods, with IntelliCorp's OOP technology, she said.

KnowledgeWare, IntelliCorp become latest CASE couple, John Pallatto, ***PC Week****, 9/2/91*

Microsoft Corp. and Hewlett-Packard Co. will cast aside their differences and announce this week an agreement to cooperate on future versions of their respective object-based technologies. Though not a formal contract, the arrangement calls for the former rivals to work together on Object Linking and Embedding (OLE) 2.0 — the second iteration of Microsoft's standard for integrating and sharing data between Windows applications that is due next year, according to officials at both companies. The pact specifies that the pair exchange technical specifications for OLE and HP's NewWave environment; it does not require them to share code, the officials said. By collaborating on object standards, HP and Microsoft will assure programmers of a single development path for OLE and NewWave, which have historically been positioned against each other. The understanding also promises that users of both environments will have a fertile application base — a factor particularly important for NewWave, which has been hampered by the lack of compatible software ...

HP, Microsoft to share object technology, Beth Freedman, ***PC Week****, 9/23/91*

Standards

... [I]t's easy to understand why the ORB submitters were so quick to cooperate by merging their individual proposals into a single entry. The ORB is complex enough without the potential problems of incompatible messages or interfaces. By agreeing to a standard interface, the vendors tore down one of the major roadblocks to object-oriented systems that cross brand-names lines. Ironically, agreeing to a standard interface didn't cost the ORB vendors their existing products or advantage of maintaining their own ways of doing things. This is a major difference from the way another industry group, the Open Software Foundation (OSF), conducts its "requests for technology." OSF examines the submitted products, then chooses a winner or merges several entries into a final selection. That's good news for the companies who win the competition — their products become instant market leaders. However, the products that don't will instantly become obsolete. Those companies must scramble to replace the competition-losing products with someone else's technology or risk being left behind. That's why OSF's competitions have generated plenty of grumbling - and even a lawsuit by one of the losers. With OMG's "just enough" approach, there's no winning product, only an interface specification. As a result, the final merged proposal supports each vendor's existing products — as well as its competitors' products. Thus, even with a standard ORB interface that all object vendors agree on, there's plenty of room for the fierce competition you normally expect among DEC, HP, Sun, and NCR. The message interface may be standard, but what's inside the black boxes is not.

Vendors willing to cooperate on object standards, Frank Hayes, ***UNIX World****, 9/91*

continued on page 19 ...

The move toward structured programming can be seen as a response to this need for greater coordination among the various parts of the program. With the functional decomposition, programmers sought to establish an orderly, hierarchical division of labor. An economic counterpart to structured programming is the bureaucratic organizing of tasks within a factory or the attempt to centrally plan an economy. While these hierarchical techniques help both programmers to manage the complexity of their programs and entrepreneurs to manage the complexity of their production processes, the ability of these techniques to adapt to changing requirements is limited. Structured programming techniques are limited primarily by their use of shared data. They modularize responsibility for action, but they retain data as a common resource. This indirect coupling of subroutines through shared data leads to software coordination failures, as data that is modified by one subroutine unexpectedly interferes with the performance of another. Here again, growing program complexity strained the ability of programmers to maintain order in their programs.

This kind of problem, arising from resources held and controlled in common, is familiar in economics. We encounter it in workshops and firms when two different workmen want to use the same tool at the same time or when the copier is tied up on an important job when something else just can't wait. We encounter it in corporations when one division needs the corporate records stored, say, by geographical region, while another division insists that they be stored by product description.

While such problems create headaches for foremen and managers, they show the fatal flaw in trying to centrally direct the wide-scale production needs of a modern economy. Technological innovations and constantly changing consumer demands limit drastically the scope of central control. Alternative methods of organization must be found to cope with the rapid changes that are part of complex economies.

OOPS: ENCAPSULATION AND PROPERTY RIGHTS

Problems of discoordination through shared data required another step in the evolution of programming practice. Object-oriented programming rooted out the problem at its source through the encapsulation of action and data. With object-oriented programming, data is not accessible at once to all parts of a complex program, but rather it is encapsulated with the routines, functions, or methods that will use it. The data is secure: objects may depend on the integrity of their own data. This helps to eliminate this important source of discoordination in much the same way that economies cope with similar problems stemming from common resources.

As Mark S. Miller and K. Eric Drexler[3] have noted, the development of object-oriented programming systems can be seen as an independent rediscovery by programmers of the virtues of property rights: encapsulation is property rights over data. They write, "encapsulation in software serves the same crucial function as property rights in human affairs: It establishes protected spheres in which entities can plan the use of their resources free of interference from unpredictable external influences."

But encapsulation gives OOPS more than simple protection from interference with data, just as property rights give economic systems more than security of possession. In addition, the information hiding aspects of encapsulation in OOPS help facilitate adaptability to changes by separating implementation from interface. This simplifies the interactions between modules, avoiding hidden dependencies. Accordingly, it enables greater system complexity because it allows modules to be modified and extended independently of other modules. The result is an essential adaptability that is prerequisite to orderliness in any complex system that must function in a changing world.

In more traditional programming systems, a client routine might tell a data storage module not only to store or retrieve a particular piece of data, but also where and how to store or retrieve it. The client routine "knows too much;" it meddles in what should be "the storage module's business." Because of this integration of interface and implementation, any changes propagate throughout the system, and coordination is lost.

There is a direct analogy here to the failure of attempts to centrally plan economies. Under central planning, the planners attempt to determine both interface and implementation, that is, they direct not only what the different plants must do for one another, but how they should do it. For example, in the planning of truck production, the planners predetermine not only that a truck plant shall make trucks, but also where they shall get their inputs of steel, tires, and glass. In turn, of course, they must also determine where the makers of steel, tires, and glass will get their supplies of ore, rubber, and silica. But this means that when more trucks are needed than expected, all the supplier industries are affected, and the whole plan is thrown into a cocked hat. The system is simply too complex for this kind of central management. Order in an economy can be restored only by decentralizing decision making. Let truck producers decide where and how they get tires and glass. Leave the information hidden from those who order the trucks.

CLIENTS AND CONTRACTS OR COMMISSARS AND COMMAND

The move to object-oriented programming, which mod-

ularizes responsibility for data along with responsibility for action, leads to a radical shift in how we think about programs. Programs are no longer seen as a sequence of actions to be performed, but as a collection of interacting components. The program, seen as a system, is not ordered around some central function. Instead, its order arises almost spontaneously out of the interaction of the objects that compose it. Similarly, whereas in a planned economy individuals' plans are seen as cogs in the great wheel of the state plan, in an open market economy the order arises out of the mutual adjustment to one another of independent agents.

Message passing between objects reflects this change. Interaction between modules is based not on following direct orders from the top, but on responding to messages, with the receiving object being held responsible for determining what action is taken. But as the name message passing indicates, with the move to more autonomous modules, a greater emphasis must be placed on the communication and interactions between the modules.

Bertrand Meyer[4] highlights this need, as well as the aptness of our economic analogy, by what he calls "programming by contract," through which these more autonomous objects are coordinated by explicitly assigning them responsibility through contractually specified interfaces. Along with ownership comes responsibility for your own actions and the need to meet contractual obligations. Rebecca Wirfs-Brock and Brian Wilkerson[5] see this responsibility-driven approach as the key to gaining many of the benefits that can accompany the adoption of object-oriented techniques. By viewing the interaction between components according to a client-server model, we can focus on the importance of encapsulation in helping us manage complexity.

Probably most significant about the growth of object-oriented programming is its promise of greater productivity through reuse and the adaptability of the software it makes possible. OOPS has meant tremendous increases in the evolvability of programs at a time when programmers were realizing that their programs were not finished when they were shipped, but needed to be maintained, adapted, and enhanced for a long time after initial "completion." This evolvability is a direct consequence of the decentralized, modular nature of OOPS.

Again, there is a strong parallel to experience with economies. Many five-year plans looked good on paper and might have worked had the world lived up to and statically maintained the assumptions they where built on. But the world was constantly changing, and the logic of central direction made adjustment virtually impossible.

MARKET-ORIENTED PROGRAMMING?

While object-oriented programming practice can be seen as incorporating two important economic institutions, property and contracts, it still does not include the most important economic institution: exchange. The next step in capturing the benefits of greater reusability and evolvability may well depend on the development of markets in software components. This is what Brad Cox has called the software industrial revolution.

The notorious problems of rational planning encountered in large-scale programming projects that were perhaps best documented in Frederick Brook's The Mythical Man Month can be addressed to some extent by evolving a software components industry that divides the programmer's task across not only different modules in the program, but also across different producers in the economy. It is here that we must look to find the answers to the challenges posed by the next level of complexity in programming. ≡≡

References and suggested readings

1. von Mises, L. *Socialism: An Economic and Sociological Analysis*, Jonathan Cape, London, 1936.

2. Hayek, F. A. "The use of knowledge in society," *American Economic Review*, 35:519 – 30, 1948.

3. Miller, M. S. and K. E. Drexler. "Markets and computation: agoric open systems," *The Ecology of Computation*, North-Holland Press, Amsterdam, 1988.

4. Meyer, B. *Object-oriented Software Construction*, Prentice Hall, Englewood Cliffs, NJ, 1988.

5. Wirfs-Brock, R. and B. Wilkerson. "Object-oriented design: a responsibility-driven approach," *OOPSLA*, 1989.

6. Lavoie, D. *Rivalry and Central Planning*, Cambridge University Press, Cambridge, UK, 1985.

FYI

Excerpts from leading industry publications on aspects of object technology

Antihype

... Ah yes, object-oriented programming. OOP is to computer magazines what dieting or Lady Di is to women's magazines. The problem with the OOP articles I've seen is that they all seem to be long on explaining how tigers are like cats and short on telling you how the heck you can use OOP in your real life. Like many, I understand OOP, but as Valentine Smith would say, "I really don't grok it"...

... [I]t would appear that any C programmer who thinks that he might find himself looking for a job in the near future had better have at least an interview-level knowledge of C++ so there is more than pure intellectual curiosity as motivation at work here...

C views: Tim's summer reading list, Tim Chase, ***HP Chronicle****, 9/91*

Approaches

... People who advocate object-oriented approaches forget that most managers are dealing with critical pragmatic problems. Often they are looking at hundreds of thousands of lines of poorly documented code written in FORTRAN, C, COBOL or some other programming language that is definitely not object oriented. Investment in "new" technology tends to take a back seat to the more basic tasks of understanding existing code and adapting it to changing requirements. The cost of changing to an object-oriented paradigm are quite correctly perceived as being very high — not necessarily in hardware or software, but rather in staff training, conversion of applications, and the impact on the organization ...

... There are some caveats to be associated with this reengineering. First, some of the pragmatic solutions will not be aesthetically pleasing to those who are well grounded in object principles. Converting a C program to C++ is likely to produce programs that are a hybrid between the two languages, meeting the requirements of the C++ compiler though not the spirit of the language. There is some historical evidence for this observation, based on the introduction of Ada. Some aerospace companies chose not to invest in Ada training, but simply gave copies of the Ada reference manual and an Ada textbook to their staff of FORTRAN programmers. While the resulting programs were legal Ada in that they were compliant with the language specification, they are best described as "ADATRAN"...

... [B]ecause they are so widely known and used it's easy to find developers "addicted" to structured analysis and/or entity-relationship modeling and many others who can shake those strongly ingrained habits with only great difficulty. One can imagine a twelve-step program for "DeMarcoholics," dealing with the painful transition to an object perspective...

Object insider, Anthony I. Wasserman, ***Object Magazine****, 9-10/91*

Applications

... "One aspect of getting data on and off PenPoint is that, with our object-oriented operating system, there are easy ways to import and export data," says [Jim] Floyd. This requires the user to merely draw the letters "E" (for export) or "I" (for import) ... "When I use the PenTOPS client on PenPoint to access data, I can select a document in PenPoint, draw an 'E' gesture to invoke 'export,' and the document is then selected with a marquee [a string of moving dots]. Once I've done that, I can just press on the document title with the pen and drag the document off to an external PC or wherever — a Macintosh or hard disk drive. PenPoint displays a list of all translation formats that application can write the resulting file out in" ...

From hands to LANs, Jim Carr, ***LAN Magazine****, 9/91*

Operating Environments

... The IBM/Apple announcement of a new desktop operating system based on object-oriented, RISC-based software could contain both good and bad news for imaging vendors and end users alike. If an object-oriented, RISC-based operating system desktop standard is achieved it ought to be a boon. However, if Microsoft uncorks its own object-oriented OS/2 (an object-oriented Windows is on the way), and the ACE consortium devises another operating system, these could plunge the computing world into confusion. There would be DOS, OS/2, UNIX, ACE (assuming the consortium accomplishes anything), and IBM/Apple AIX-based systems to choose from. The problem would then become a dilemma of choices. Confusion for everything connected with it, including imaging systems based on different platforms ...

The empire (IBM) strikes back (at Microsoft and the ACE consortium), ***Imaging Technology Report****, 8/91*

Databases

... To support its focus on strong data-access and data-modeling capabilities, Borland has created new cursor technology. It allows users to draw a database schema on screen while automatically programing the links between the database files. Without any programming, users will be able to create complex one-to-many relationships graphically and to consistently maintain links between database files and tables. The technology will be incorporated into the object-oriented Paradox for Windows, due in about seven months ...

Borland outlines plans for client/server thrust, Karen D. Moser, ***PC Week****, 9/23/91*

```
        ifTrue: [
            aStream nextPutAll: ';
    " ', ea printString, ' "
    addWidget: ('.
        (ea at: 5) = #text
            ifTrue: [
                aStream nextPutAll: 'StaticText new
        yourself;
        string: "', (ea at: 2), "';
        font: f;
            justification: #right;
            size: ', (self displaySize: (ea at: 3) as: text)
                printString]

            ifFalse: [
                n := ea at: 2.
                    aStream nextPutAll:
                ((n includes: $: )
                    ifFalse: [n ,' := ']
                    ifTrue: ['(', n,' put: ('] ) ,
                ((ea at: 5) = #output
                    ifTrue: ['StaticText new']
                    ifFalse: ['EditBox new ']),
                ((n includes: $: )
                    ifFalse: ['
            yourself; ']
                    ifTrue: [')')']),'
```

```
                nameForInstVar: "', n, "';
                font: f;',
                    ((ea at: 5) = #output
                        ifTrue: ['
                justification: #autosizedWrapping;']
                        ifFalse: ["]), '
                size: ', (self displaySize: (ea at: 3) as: (ea at: 5))
                    printString].

                aStream nextPutAll: ')
            position: ', (self location: ((ea at: 1) at: 1) @ ((ea at: 1)
                at: 2)) printString]].

    aStream nextPutAll: ';
    size: ',
        (self screenSize) printString; nextPutAll: '!!'.
    (0 @ 0) = location
        ifFalse: [
            aStream nextPutAll: '

windowLocation
        "Position the popUp."
    ^',((self location: (location x - 1) @ (location y - 1))
printString), '!!'].

    aStream nextPutAll: ' !!'; cr.! !
```

worth the time and expense of reimplementing the prototype closer to the metal in some language like C++, PL/I, or COBOL. Good enough is often good enough.

Using a package like Widgets/V has some additional benefits. The Widgets application windows generated are portable between IBM and Macintosh environments. Portability will soon be available to OS/2 Presentation Manager and Windows 3.0 environments. ≡≡

References and suggested readings

1. IBM Corp. *Systems Application Architecture – Common User Access: Basic Interface Design Guide*, IBM Corp., New York, SC26-4583-0.

2. IBM Corp. *Systems Application Architecture – Common User Access: Advanced Interface Design Guide*, IBM Corp., New York.

3. Rovira, C-A. Rovira diagrams, *Computer Language Magazine*, 7(1), 1990.

4. Digitalk Inc. *Smalltalk/V for PCs, ATs, Macintosh, Windows 3.0, OS/2-PM*, Digitalk Inc., Los Angeles, CA.

5. Acumen Software. *Widgets/V for /V286 and /V Mac*, Acumen Software, Berkeley, CA.

... continued from page 11

of the original C++ standards. However, its power is recognized and it has been included in version 2 of the C++ standards. Borland has included it in their new C++ compiler. Now, at last, I can find the things I want to reuse.

As far as understanding the program goes, the concept of methods really means writing short, independent modules. The brevity should allow me to understand the methods I want to inherit. And, if it doesn't, I can still write the one I need and inherit all the methods that don't require change.

Finally, and probably most powerful, when in the course of human events it becomes necessary to modify a method (yes, it is still possible to create code with bugs), the code is changed only once, in the original class. All the inheriting subclasses automatically access, without recompilation, the corrected code!

Now the question that remains is what to do about OOT? I, for one, have bought in, and here's why:

- New vocabulary is always a pain to get used to. But, as with structured design, it helps me understand and discuss previously fuzzy ideas in a more precise fashion.
- A new design approach gives me one more perspective on the design process. I don't believe that any one design method is right, but, the more methods I know, the cleaner my design ultimately is, and the more comfortable I am with it.
- The new tools make reuse practical and maintenance easier and safer. ≡≡

CHOOSING OBJECT TECHNOLOGY ≡≡

What's the object?

by Sig Rosenthal

I wrote my first program in 1958. It was terrible by today's standards. It was probably terrible by the standards then, too.

Since then, there have been many techniques to improve software development (modular programming, structured design, etc.). Each of them eventually improved the state of software development and today they are considered the norm. Each of them went through the same tortuous cycle of acceptance before they reached their current status. The usual acceptance cycle went something like:

1. What is this new thing, anyway?
2. Why should I use it, it's just a nuisance?
3. Oh, I know all about it, I've always written code that way. I just never had a name for it.
4. Now that I really understand it, I see that it is new, different, and better. And I can rationally decide where and how to use it.

Well, the latest wave is upon us. It's called "object-oriented" something. As in object-oriented programing (OOP), or object-oriented databases (OODB), or object-oriented programming systems (OOPS). For the rest of this article, I'm going to call it object-oriented technology (OOT) because it is really useful only with the support technology that automates some of its functions. Where appropriate, I'll point out the difference between the concept and the implementation.

The object of OOT is to make it easier (cheaper, faster) to create and maintain high-quality software. This isn't any different from the object of all the previous techniques. Although it seems to conflict with some of the earlier concepts (like functional cohesion), I think it is an extension of some existing good practices that are becoming better defined. (But then I am only in phase 3 of the acceptance cycle.)

Before I discuss the object(tive) of OOT, let's start with some basic definitions. Despite the strange names (remember coincidental or communicational cohesion?), the concepts are simple and have been around a while. First, the object is at the center of all this.

An *object* (as in object-oriented) is a package of software routines and data that represents a real-world thing. The routines are typically called *methods* (sometimes *member functions*). That's a rough description and hopefully not to object(ional) to the purists. In practice, the "real-worldliness" can be pretty abstract. An integer or a linked list might be used as an object.

At first this description doesn't seem to differ too much from a traditional program with its associated data files. But the emphasis is decidedly different. Rather than associating the data with a program (or vice versa), the object is considered as a whole. The data and methods are inextricably bound together, one doesn't exist without the other. An object's data can only be accessed through its methods.

A *class* is an abstraction of a group of like objects. Class and object are frequently used interchangeably and the meaning is usually clear from the context. Strictly speaking, however, an object is an instance of a class. For example, integer is a class and 1 is an object (it is an instance of an integer). An integer has certain properties. For example, you can add two of then together and the result is always an integer. Since 1 is an integer, it automatically has this property. You can add any integer to 1 and the result is always an integer. A class describes a type of object. An integer is a type of number.

This concept is not new. All computer languages have data types and operations defined on those types (Integer - ADD; Real - ADD; Boolean - OR; String - CONCATENATE; etc.). OOT's classes are an exten-

Sig Rosenthal is a consultant living in Rye, NY. He has over 25 years' experience developing software. He has worked on mainframes, minis, and micros.

sion of this concept of types and operations. You can define a class, savings account for example, and specify the operations that can be performed on it (OPEN, DEPOSIT, WITHDRAW, etc.).

A *message* is the mechanism that object use to communicate with each other. Since one object cannot modify another's data, this is the process used to get any work done. For example, a teller object cannot directly modify a savings account's balance. The teller must send a message to the appropriate savings account to deposit money. The accounts DEPOSIT method will then update the balance. Again, the concept of a message isn't new (though its vocabulary is), it is simply the "calling" mechanism used to coordinate separate processes.

Encapsulation is a fancy way of saying that an object's data can only be accessed by its own methods. No other code is allowed to modify or even see the data. You can think of the methods as a sort of protective shell, completely wrapped around (encapsulating) the data. In pre-OOT days this was called *data hiding*. It is pervasive in database management systems (DBMS), such as IMS, where a program would never manipulate data directly but only through calls to the DBMS. The DBMS would do the actual GET or UPDATE and maintain all the data as well as the pointers and/or indices, freeing the programmer from these tedious and error-prone activities.

Inheritance is the mechanism for creating new classes out of old. The new class is said to "inherit" the methods and data structures of its parent. The purpose of inheritance is the efficient reuse of code. For example, if you have a savings account class, you can define a certificate of deposit as a descendant class. It will inherit all of its parent's methods (OPEN, DEPOSIT, WITHDRAW, etc.). Then you can change the DEPOSIT method to define the maturation date of the certificate and the WITHDRAW method to impose a fee for an early withdrawal. All the rest of the (thoroughly debugged) code is used, as is, from the parent class. I'm sure you've all done this before. Create a new program by taking a working one and modifying it slightly to satisfy a new need. It's not a new concept but it is a powerful one. Now it's an easy one and safer. Safer because the unchanged code isn't copied, the original code is reused.

Since the savings account (and a checking account) may be a descendant of a more general class, bank account, it is apparent that the classes themselves form a hierarchy. This is usually referred to as a class hierarchy.

Polymorphism is probably the simplest OOT concept with the fanciest name. It means "many forms" and its intent is to allow the same operation (method name) for many forms. For example, assembler language might have many different ADD instructions: ADD INTEGER, ADD PACKED, ADD DECIMAL, ADD FLOAT, ADD REGISTER, ADD LONG, ADD DOUBLE, etc. Polymorphism means that the same method name, ADD, should apply to all forms of the object (all the classes in the hierarchy). And each object, knowing its own internal structure, will correctly perform the add.

Persistence is the last definition we need. When an Object-oriented system is running (don't think of it as a program, but a whole system), new classes may be defined, and objects created and modified. When the system terminates, all those things disappear. It might, on occasion, be useful to have some of those objects (like the savings account) stick around until the next time the system is run. Objects that continue to exist even when the system isn't running are "persistent." (You might recognize this concept as an extension to saving data in a file so it can be found again later.) In the OOT world, since an object is data and methods, this means that any update to an object's data will automatically execute the object's update methods which, hopefully, have all the required validation logic built into them. No careless programmer can forget to validate an object's data when updating it! (Don't worry, the actual implementation doesn't really store all of the code with each object. Disk vendors have been disappointed.)

Well, that's it. Some new vocabulary, some old or enhanced ideas, and a new way of looking at it all. Now to see what all the fuss is about.

What does OOT really offer? Basically, two things. An additional approach to designing systems and a way of reusing code.

Just as Constantine's structured design provided a method to handle transaction-based systems that were clumsy with Jackson's design method, so does OOT provide an approach to systems that are clumsy with current techniques. For example, SNA, with its multitude of cooperating processes grouped into LUs and residing in nodes throughout a network, would lend itself very nicely to an OOT design. In fact, the whole class of systems called client/server or cooperating programs would fit nicely into the OOT view. In addition, the fancy graphics and user interfaces showing up in the PC world are almost impossible without an OOT frame of mind.

It's worth comparing the orientation of an OOT design to current designs. The most popular current methodology is made up of data flow diagrams and structure charts, often in conjunction with entity/relationship diagrams.

The E/R diagrams are static. They depict the relations between different entities in the business. The objective of E/R diagrams is a stable definition of all the data in an enterprise. This allows all the application systems to have a consistent view of their data, allowing easier sharing of the data and easier creation or modification of the application systems.

Structure charts are also static. They depict the control relationship between software modules, defining the allowable (and preplanned) passing of control between


```
            file nextLine]].
    tmp := Array with: (Array with: y with: x) with: nam
        with: len with: typ with: stt.
    stt = #text
        ifTrue: [
            nam := nam storeString.
            nam := nam copyFrom: 2 to: nam size - 1.
            (pt := pat match: nam index: 1) notNil
                ifTrue: [
                    size := x @ (y + len - 1).
                    str := (nam copyFrom: pt y + 1 to: nam
                        size) asStream.
                    tmp at: 2 put: (nam copyFrom: 1 to: pt x - 1).
                    x := (str nextWord) asInteger.
                    "skip '."
                    str position: str position + 1.
                    y := (str nextWord) asInteger.
                    location := x @ y].
            fields add: tmp]
        ifFalse: [
            cnt := fields addi: tmp.
            variables add:(tmp at: 2) with: cnt]!

processHeader
        "Answer an array with the number of regions and
        fields to set the class name."
    | lin rgns flds |
    lin := file nextLine.
    variables := CrossReferenceDictionary new.
    fields := SortedCollection sortBlock:
    [:a :b |
        (((a at: 1) at: 2) * 80 + ((a at: 1) at: 1)) <
        (((b at: 1) at: 2) * 80 + ((b at: 1) at: 1))].
    class := ((lin copyFrom: 1 to: 32) trimBlanks)
        asClassName.
    rgns := (lin copyFrom: 75 to: 76) asInteger.
    flds := (lin copyFrom: 77 to: 79) asInteger.
    lin := file nextLine.
    ^Array with: rgns with: flds!

processRegion
        "We don't care about those. Just consume the line."
    | lin |
    lin := file nextLine.
    ^nil!

resolveRepeats
        Find duplicated field names to handle repeats."
    | val cnt |
    variables associationsDo: [:ea |
    (val := ea value) size > 1
            ifTrue: [
                cnt := 0.
                val asSortedCollection do: [:ep |
                (fields at: ep)
                    at: 2
                    put: (
                        ((fields at: ep) at: 2),
                        ' at: ',
                        ((cnt := cnt + 1) printString))]]]!

screenSize
        "Answer a point containing the extent of the
        screen."
    | answer |
    answer :=(8 * (size y)) @ (12 * (size x)) + 50.
    ^answer!
```

```
writeClass: aClassName instVars: aXRefDict on: aStream
        "Write a class def for aClass."
    aStream cr; nextPutAll:

'Cdfs3270 subclass: #', aClassName; nextPutAll: '
    instanceVariableNames: "'.
        aXRefDict associationsDo: [:ea |
            aStream nextPutAll: ea key, ' '].
        aStream nextPutAll: "'
    classVariableNames: ""
    poolDictionaries: ""!!

!!',aClassName, ' methods !!
initialize
        "Fill fields with something."
    | tmp |
    tmp := "..........".'.
        aXRefDict associationsDo: [:ea |
        ea value size = 1
            ifTrue: [
                aStream
                    cr;
                        nextPutAll: ' ', ea key, ' string: tmp.']
            ifFalse: [
                1 to: ea value size do: [:i |
                aStream
                    cr;
                        nextPutAll: ' (', ea key, ' at: ',
                            (i printString), ') string: tmp.']]].
    aStream nextPutAll: ' !! !!' ;cr!

writeCreateWindow: fields for: aClassName on: aStream
        "Write a createWindow method for aClass. fields is
        an OrdferedCollection of (1) field y, x positions,
        (2) name (type C or P) or text (type nil), (3)
        length, and (4) type ($C, $P, or nil)."
    | x y l n s |
    textFont := Font getFont: 'eightLine'.
    editFont := Font getFont: 'eightLine'.
    aStream cr; nextPutAll: '!! ', aClassName, ' methods !!

createWindow
        "Define a window for a 3270 type screen."
| f |
    f := Font named: "eightLine".'.
    "Handle any repeated fields."
    variables associationsDo: [:ea |
    (s := ea value size) > 1
        ifTrue: [
            aStream nextPutAll: '
    ', ea key, ' := Array new: ', (s printString), '.' ]].
    "Resume window creation."
    aStream cr; nextPutAll: '
    ^TitledWindow new
    yourself;
    title: " ', aClassName, ' " ;
    closable: true'
    (0@0) = location
        ifTrue: [
            aStream nextPutAll: ';
    menuBar: (
        MenuBarWidget new
            yourself;
                items: self menuDefinition
    )'].

    fields do: [:ea |
        ea notNil
```

matical representation and specification of dialogues,[3] and these techniques are being integrated into a tool for the analysts to use, the previous and current technique consisted of descriptions written on PCs in WordPerfect.

The next step involved taking the written description of the dialogues, parsing them, and generating Smalltalk code based on the information contained in these files.

The prototyping system developed is also capable of taking COBOL or PL/I record layouts from Exceleretor or lifted from program code and generating Smalltalk objects for direct use in unit record files or SQL structures.

WHAT DOES MIS WANT?

It is possible to use output from CASE tools to generate Smalltalk code directly.

Using upper-CASE tools to aid in analysis and design speeds up the specification of end user applications. Using Smalltalk to interpret these specifications and generate code that it can use in an implementation of the application can speed up the process of prototyping. Once a complete prototype is up and running, the organization can examine its performance metrics, estimate some capacity plan, and evaluate whether it would even be

Listing 1. The Exceleretor Screen Definition class

```
Object subclass: #ExcScrDef
    instanceVariableNames:
        'file class variables fields textHolder textFont
        editFont location size '
    classVariableNames: ''
    poolDictionaries: ''

!ExcScrDef class methods !

generateClass
        "shift to instance and doIt."
    self new generateWidgetWindow! !

!ExcScrDef methods !

displaySize: aLen as: aType
        "Answer a point that corresponds to the extent of
        aLength. aType is no lenger used since the
        typeFace was decided on."
    | answer |
    answer := 9 * aLen @ 12.
    ^answer!

generateWidgetWindow
        "This gets complicated. First get a file (SFReply)
        and scan it to extract three things. (1) Header.
        This will become the name of a class that will be a
        subclass of class Cdfs3270. It also specifies the
        number of regions and fields. (2) Region
        description. We don't care about these. (3) Fields
        descriptions that provide the position, size, and
        content of a display, input or output, data
        element (also include variable names)."
    | tmp |
    "Get a file"
    file := SFReply getFileExt: '.map'.
    file isNil
        ifTrue: [^self].
    CursorManager execute change.
    location := 0@0.
    size := 24 @ 80.
    textHolder := WriteStream on: ''.
    tmp := self processHeader.
    1 to: (tmp at: 1) do: [:j |
        self processRegion].
    1 to: (tmp at: 2) do: [:j |
        self processField].
    file close.
    self resolveRepeats.
    self writeClass: class instVars: variables on:
        textHolder.
    self writeCreateWindow: fields for: class on:
        textHolder.
    "Compile and evaluate the generated text (in change
    log format)."
    (ReadStream on: textHolder contents) fileIn.
    CursorManager normal change.
    (Smalltalk at: (class asSymbol)) open!


location: aPoint
        "Answer a point that corresponds to aPoint."
    | answer |
    answer := (aPoint x - 1 * 8) @ (aPoint y - 1 * 12).
    ^answer!

processField
        "Process two to five lines. fields is an
        OrdferedCollection of (1) field y, x positions, (2)
        name (type C or P) or text (type nil) (3) length,
        and (4) type ($C, $P or nil)."
    | cnt lin pat pqu x y len stt nam typ tmp pt str |
    lin := file nextLine.
    pat := Pattern new: '@'.
    pqu := Pattern new: ''.
    x := (lin copyFrom: 1 to: 2) asInteger.
    y := (lin copyFrom: 3 to: 5) asInteger.
    len := (lin copyFrom: 10 to: 11) asInteger.
    stt := (lin at: 6).
    stt = $T
        ifTrue: [
            typ := nil.
            stt := #text].
    stt = $I
        ifTrue: [
            stt := #input].
    stt = $O
        ifTrue:[
            stt := #output].
    stt = #text
        ifTrue: [
            nam := file nextLine trimBlanks storeString.
            nam := nam copyFrom: 2 to: nam size - 1.]
        ifFalse: [
            file nextLine.
            nam := ((lin := file nextLine)
                copyFrom: 1 to: 32) trimBlanks
                    asInstVarName.
            typ := lin at: 35.
            stt = #input
                ifTrue: [
                    file nextLine.
```


modules. They allow an impersonal assessment of just how closely related (tightly coupled) modules are.

Data flow diagrams are not quite as static as the E/R diagrams or structure charts. After all, they do show the "flow" of data through a system. However, DFDs aren't very dynamic. They only show a "steady state" status of a system, not the initiation or termination of a system. And they are usually described as showing the static "pipes" through which the data flows rather than the data itself. And worse, data doesn't really flow through most systems. It sits there until some program comes along, picks it up, manipulates it, and sets it back down.

Don't get me wrong. I really like these methods. I've used them for a number of years, and life is really much easier than it was before them. What I want to point out is the different view provided by OOT. I see two major differences in the design philosophy. One is the approach to the dynamics of a system; the other is the approach to the packaging.

The first major difference is that OOT design focuses on the dynamics of a system. The objects are the basic building blocks. (That's why the object definition emphasizes equally the data and methods.) But the system design centers on messages that are the interaction of the objects. Customers *open* accounts. Tellers *accept* payments. Vendors *send* bills. Stock is *withdrawn* from inventory. OOT designs bring out the dynamics of a system as does no other methodology does.

The second major difference is the consideration of packaging. With the current methodologies, the approach is to define independent functions — those with high cohesion and low coupling. The decision of which functions should be packaged together is left till the end of the design activity and is usually driven by performance (size, speed) constraints. This conflicts with the earliest design activity, architectural design, which identifies the portions of the system (functions?) that are separable (not packaged together?). In effect, we are making the same decision at the very beginning and very end of the design activity. Yes, the architectural design is at a "high" level and the packaging design is at a "low" level, but the separation between levels is a little blurred and the decisions are very similar.

In contrast, OOT design focuses on the packaging from the very beginning. The initial activity, identifying objects, is by its very nature defining the packaging — which functions (methods) and data should be inextricably bound together.

Having identified the major differences in the design process, we can now look at the basic approach to creating OOT designs:

1. Define a class (or class hierarchy) to represent things in the real world that are of interest in the system being built. (Various bank accounts, for example.) I haven't built any real OOT systems yet, but I think that the basic entities of a business would be a good place to start looking for a systems objects.
2. Define the data needed to support an object of the class and the methods that will manipulate the data.
3. Define similar classes that will use some of the original classes data and/or methods.
4. Define the messages that will pass between methods. These methods may be in the same or other objects.

After the design is complete, all that's left is writing the code to create the classes, define the data, and run the methods. A SMOP (Simple Matter Of Programming) if ever there was one. Of course, if you're not using Smalltalk, C++, or one of the other object-oriented languages, it is a bit more difficult. But at least your basic design is done.

As far as the second advantage of OOT systems, reusing code, that's an idea whose time has finally come. Programmers have been trying to reuse code for years and have succeeded only in a limited way. The limitation has been caused by the lack of appropriate tools. I can often reuse code I have written recently (in the last year or so). But it is difficult to reuse someone else's code for two reasons. First, it frequently would take longer to find the code than to write it myself. Second, even if I could find the code, if it required any modification it would surely take longer to figure out the modifications than to write it myself. OOT addresses both of these problems. The first is solved by a tool called a browser. This, to my mind, is what makes OOT really practical — without this support technology OOT is just another cute idea.

A browser (or class browser) is a program that allows you to move through a class hierarchy, finding all the descendants or ancestors of a class and the data and methods of each.

Now, suppose I had to write a program to support NOW accounts. I'm not sure if such code already exists, so I can look in a class hierarchy for a NOW account. Actually, since I'm not sure that such a class already exists, I'll look for one of its ancestors. I don't know if it's descended from a savings account (it pays interest) or a checking account (you can write checks on it), so I'll just go into the hierarchy to find a generic bank account. I can then use the browser to go down the hierarchy and, if a NOW account is already defined, I'll find it. If not, I'll be able to create one by inheriting (reusing) the appropriate methods from the checking and savings accounts. Inheriting from more than one class is called *multiple inheritance*. Here is one place where the (excellent) concepts have outpaced the implementation. Not all OOTs support multiple inheritance. It was not part

continued on page 16 ...

MIS

OOP: the MISsing link

by Charles-A. Rovira

The interest MIS shops have in object-oriented programming has grown almost exponentially over the past few years.

However, these same organizations have been struggling to find ways to use object-oriented techniques and programming without having to throw away all of the tools that they have acquired in their continuous efforts at modernization.

This article describes part of a tool, implemented in Smalltalk, that allows Smalltalk to use screen layouts created by analysts using a package they knew and were comfortable with, namely Exceleretor. This tool generates Smalltalk/Widgets application windows without requiring any coding by the analysts or programmers, namely the author and his colleagues, whose agendas and schedules were not under control.

We'll examine some means used to translate screens from Exceleretor to a Smalltalk/Widgets combination that would meet our requirements. Later, the work was expanded to include the interactive definition of dialogues.

WHAT DOES WINDOWS WANT?

Most analysts who only have MIS experience find themselves faced with an exasperating problem. Fortunately, IBM provides an answer for analysts in two books: *Systems Application Architecture – Common User Access: Basic Interface Design Guide*[1] and *Systems Application Architecture – Common User Access: Advanced Interface Design Guide*.[2]

These two volumes cover the entire range of what IBM wants to see in applications implemented for their machines as well as for other machines. While this might seem a bit excessive as a corporate goal, these books are well written and cover most of the information likely to be needed for a very broad range of applications. They are quite likely to succeed in spreading Systems Application Architecture – Common Access User (SAA-CUA) compliance across the face of the computing world in an astonishingly short period of time.

Smalltalk's windowing environment

Originating in the early 1970s and coming into full bloom in 1980, Smalltalk was the first language and environment that had windows, icons, mice, panes to contain text, lists of text items, and bitmapped graphics. Unfortunately, it stopped there, and the developer was left to refine the environment into whatever an application might require.

While, or perhaps because, the process was not overly difficult, it resulted in a plethora of incompatible interfaces. Not to cast aspersions on the X Window System, OpenLook, or Motif, the coming of the Macintosh, IBM's subsequent development of Presentation Manager, and Microsoft's Windows efforts have enforced some de facto standards and consistency across environments.

Widgets' windowing environment

Smalltalk's windowing environment operates at a transaction level, while most applications need to work with fields, discrete objects within a transaction that present or acquire information relevant to the instantiation of the transaction. What is needed is a tool that allows the creation and manipulation of fields within a pane.

Widgets provides just such a tool. With Widgets, a window can be bedecked with any number of a wide variety of fields. These fields can display images, text, numbers, pull-down menus of values for an item, lists of text items in which the user can select one or more items, and palettes of graphic images. Windows can also have menu bars, hierarchically organized menu items with accelerator keys defined for keyboard selection of items and their related actions.

The only possible shortcoming of Widgets concerns the behavior of using certain keys, namely F10 and the cursor movement keys, for activating the menu bar, moving across menus, pulling them down as they are selected, and selecting items in the visible menu. To be fully CUA compliant, the package requires some additional code, but this code would require only a rearrangement of the services that Widgets provides rather that the creation of entirely new code.

Application windowing requirements

Most mainframe applications don't require the sophistication that Widgets can bring to the user interface. That may be because most MIS analysts have yet to round the learning curve that would lead them to demand more out of their user interface than is available with their 327x terminals. Their CASE tools also show the same mind-set.

For the present, a straight translation of 327x screen layouts to Widget windows is sufficient to satisfy the needs of the analysts.

WHAT DO ANALYSTS WANT?

The actual process of transforming a screen description into some executable Smalltalk code is admittedly not very complicated. It does, however, require in-depth knowledge of Widgets, Smalltalk, and various vagaries of fonts. Needing to know about fonts always comes as a surprise to MIS analysts who have never had to deal with them in the fixed-format, single-font world of mainframes.

Like everything else, Smalltalk code is executed in a certain context. It resides in a Smalltalk image that is basically concerned with providing prototyping services to analysts.

To make the process as simple as possible, an application window is on-screen, and the conversion process is initiated by pressing on a button on the window. After selecting a file that contains the desired Exceleretor layout and a few seconds of staring at a cursor shaped like a watch, the next visual clue to the analyst is the appearance of the window on the screen.

When the prototyping application terminates, the analyst is prompted about whether he wants to save the changes; the application either saves itself or doesn't, depending on the reply.

ExcScrDef

While the code contained in Listing 1 is the actual code that accomplishes the Exceleretor-to-Smalltalk/Widgets translation, there is a good deal of auxiliary code that is required to accomplish the translation. CrossReferenceDictionary was a class developed to do what its name says, provide a cross-reference of keys and their use. SFReply, along with some Widget application windows, was developed to provide a mechanism that enhanced DOS and to provide compatibility between DOS and the more sophisticated Macintosh operating system.

The process is begun by sending the class ExcScrDef the message generateClass. This creates an instance of itself and tells it to generateWidgetWindow. This method then asks SFReply to getFile from the user looking only at files with extension .MAP. This window works like the Macintosh StandardFileDialog window, but watch out for the look-and-feel lawsuits. The user can cancel or select a file. Canceling exits immediately. The selected file will be scanned to retrieve a header that specifies the name of the screen, and the name that the resulting Smalltalk class will be given. This process is described in the method processHeader. Should any regions have been defined for the screen, these are ignored by processRegion. The field definitions are gathered in processField, and the file is closed.

The next step, resolveRepeats, was required because we were defining repeating groups for their screens in Exceleretor by copying groups of lines and their definitions and pasting them wherever we wanted them on the screen. This yielded layouts with the same variable name occurring in however many lines had been copied and pasted. It was therefore necessary to keep a CrossReferenceDictionary that associated a variable name with the line or lines on which it had been used.

Once possible repeats had been handled, it was only necessary to write a class definition in a text holder — accomplished by writeClass:instVars:on: — followed by window creation methods in a format that Widgets could use — accomplished by writeCreateWindow:for:on:. After that, the contents of the text holder were compiled by fileIn, and the window was brought up on the screen.

It might be worth returning to processField for a moment. The fields were collected as they occurred in the file but they were kept in a SortedCollection, which maintained their order on the screen.

The author will admit that he was in a hurry in developing the class. For the sake of expediency, he neglected to create an interim object, which he might have called ExcScrDefFld, to describe each field and settled for keeping each field in an array of four or five objects. This led to a definition of the sortBlock for keeping the fields in order of their position in the layout to look like:

```
fields := SortedCollection sortBlock:
[:a :b |
  (((a at: 1) at: 2) * 80 + ((a at: 1) at: 1)) <
  (((b at: 1) at: 2) * 80 + ((b at: 1) at: 1))].
```

Extending the mechanisms

This article examines only a part of the code developed to translate Exceleretor into Smalltalk. Screen layouts are an important component, and translating them to Widgets was a good first step. However, this describes only the surface behavior of the application. Windows present information within a context. They occur in some form of dialogue.

While the author has written articles on diagram-

Now based in Ottawa, Canada, Charles-A. Rovira has been involved with data processing since 1975 and with Smalltalk and other object-oriented technologies since 1987. His CompuServe ID is: 71230,1217. He'll admit to some unusual literary influences, such as Douglas Adams, Terry Pratchett, and D. H. Lawrence.

≡ Your Turn ≡

How about it? How are we doing? This is the place to send us your message. Tell us what you like and what you don't, what you'd like to see but don't — or whatever other feedback you'd like to send. Send comments to Marie A. Lenzi, 155 E. 31st St. #22, NY, NY 10016; fax to THE HOTLINE, *(212)725-0224; or, if you're ordering, send along with your order.*

Please check whichever box applies:

	Read entirely	Scanned	Found helpful	Will refer to	Didn't read
Feature	❑	❑	❑	❑	❑
Getting Started	❑	❑	❑	❑	❑
Multimedia	❑	❑	❑	❑	❑
FYI	❑	❑	❑	❑	❑
Book Watch	❑	❑	❑	❑	❑
Product Announcements	❑	❑	❑	❑	❑
Calendar	❑	❑	❑	❑	❑

Please rate the following **1** (dislike or strongly disagree) to **10** (like or strongly agree):

Presentation of material __ Comments: _____

Accuracy of material __ Comments: _____

Overall helpfulness in your job __ Comments: _____

What else would you like to see in THE HOTLINE? _____

Any other comments? _____

_____

_____

Name, title & phone (optional): _____

hotline on
OBJECT-ORIENTED
technology

❑ **Yes,** plug me into the latest thinking and developments in object-oriented technology. Enter me as a subscriber at the term marked below and rush me the current issue. This is a risk-free offer — I may cancel my subscription at any time and promptly receive a refund for the unused portion.

1 year (12 issues) ❑ $249
(outside US add $30 per year for air service)

2 years (24 issues) ❑ $478 *(save $20)*

Back issues @ $25 each *($27.50 foreign)*:
Vol.1, Nos. _____
Vol.2, Nos. _____ Vol.3, Nos. _____

❑ **Phone order**
Call Subscriber Services
(212)274-0640

❑ **Fax order**
Fill out term and name/address information, then fax to:
(212)274-0646

❑ **Send me a copy of**
THE INTERNATIONAL OOP DIRECTORY
@ $63 ($71 foreign)
(all directory orders must be prepaid; fax or mail credit card information to SIGS, 588 Broadway, #604, NYC 10012; make check payable to OOP Directory. Foreign orders must be prepaid in US dollars drawn on US bank.) D1MC

❑ **Bill me**
❑ **Check enclosed**
❑ **Credit card orders**
❑ MasterCard
❑ Visa

Mail (and make out check, if applicable) to:
THE HOTLINE
Subscriber Services
P.O. Box 3000, Dept. HOT
Denville, NJ 07834
(foreign orders must be prepaid in US dollars drawn on a US bank)

Card# Expiration Date
Signature
Name
Company
Street/Building#
City/Province
ST/Zip/Country
Telephone D1MC

Send me a complimentary copy of your related publication(s):

❑ THE JOURNAL OF OBJECT-ORIENTED PROGRAMMING

❑ OBJECT MAGAZINE

❑ THE C++ REPORT

Call Subscriber Services
(212)274-0640
or send order to
SIGS, 588 Broadway, #604
New York, NY 10012

D1MC

hotline on
OBJECT-ORIENTED
technology

VOLUME 3, NUMBER 2 THE MANAGER'S SOURCE FOR TRENDS, ISSUES & STRATEGIES DECEMBER 1991

Accepting object technology

It was early 1989 when I first began to seriously develop my interest in object technology.



Lydia Bennett

At Digital, I had moved from managing products in office systems to working in the object space. My first product was DEC Trellis, an object-oriented application development system. I was also chartered to develop market and product plans for an object-oriented database system.

I had the privilege of being assigned a team of developers who had many years of experience in object technology. They knew the lingo, could explain objects to a great technical degree, and had worked on the DEC Trellis prototype in one of our research groups. While I found each team member to be informative I also found them to be so close to the technology that they explained it at a level too esoteric for me. I really didn't understand the tie between object functionality and the benefits our customers were hoping for. The object definitions to date were mysterious and ethereal.

LEARNING ABOUT OBJECT TECHNOLOGY

This presented me with a challenge: trying to define what an object was so that basically anyone could understand it. I realized that if I couldn't grasp the concepts quickly with all my years in the computer industry, how could a high-level business manager, one who might be responsible for funding projects, understand it. I had already seen that some influential individuals were not sure they comprehended what was said about the technology and, therefore, were unclear as to the amount of investment that should be allocated.

Learning about objects required more effort than I previously assumed. It takes much reading and questioning of the experts. To understand it as fully as possible means to engross yourself in meaningful materials on a regular basis. It means talking to those in the know and being able to validate your understanding with, preferably, an implementor of the technology. I was very lucky at Digital to have quite a complete library as well as implementors to query. My goal was to be able to provide that same class of educational support to any internal employee and to our customer base. It seemed everyone was asking, "What is an object and why should I care?"

continued on page 3 ...

IN THIS ISSUE

FEATURE — Accepting object technology *Lydia Bennett*	1	2	FROM THE EDITOR
GETTING STARTED — Adopting objects: a path *J. Michael Connell*	5	15	PRODUCT ANNOUNCEMENTS
MULTIMEDIA — Incorporating graphical content into multimedia presentations *Louis Berman & Kenneth Weiss*	9	17	NEW PARTNERINGS & ACQUISITIONS
FYI — Industry publication excerpts	12	19	CALENDAR
BOOK WATCH	14	20	YOUR TURN

From the Editor



My company, Syrinx, went into business in 1987 to build a CASE product. Based on our initial study, we knew that to get our CASE product on the market quickly and to be responsive to the market once we got there, we had to implement it in object technology. In '88 there weren't a lot of objects on the market. We looked around at the available tools and the state of the art and quickly discovered that our business plan needed significant revisions. It was apparent to us that object technology was definitely the wave of the future. For complex software to be built quickly, to be flexible, and, therefore, to be cost effective in the next decade, we felt that this emerging object technology had to be the correct direction. We based this on our twenty plus years each in the industry and on the growth, directions, and outcomes we had lived through. The CASE product quickly went on the back burner, and we sailed head-long into object technology. That was our first response to objects, and over the years I have encountered many, many others who have had the same initial response.

Some people look at object technology and instinctively know that this has got to be the way to go. Many others, however, are not so reckless. They require much more study and convincing that a major new technology is right for their environment. The cost, of course, is the most significant factor; however, there is also the issue of being "wrong." What if my perception is incorrect and therefore my decision to adopt this new technology is an unsound, expensive decision? What happens to my company? What happens to my job? These are generally not easy decisions to make and cannot be made quickly or in a vacuum. The decision to adopt objects is one that should be based on education, financial modeling, technology trials, and a long hard study of the corporate requirements, resources, and strategy. A team with a solid cross-section of skills should be assembled to examine these issues. Implementing object-oriented programming languages can be thought of as one of the easy parts. One of the tough parts is convincing first yourself and then your organization that object technology is the way to go. Even the most technologically sophisticated organizations have to be sold on the new technology. In "Accepting object technology," **Lydia Bennett**, Strategic Marketing Manager for Emerging Technologies, talks about the "selling" of objects within Digital Equipment Corporation.

A fine compliment to Lydia's piece is the piece from **J. Mike Connell**. Michael Connell has been working with object technology for quite a while, first with Servio Logic and then with Instantiations. In his time, he has seen many organizations' approaches to implementing object technology. In "Adopting objects: a path and pitfalls," Mike offers us a sane approach for first exploring and then implementing object technology in large organizations. This approach and the accompanying pitfalls are based on his experience with organizations who have tried to implement object technology—some with varying degrees of success; others with, well...

I am pleased to have **Kenneth Weiss** and **Louis Berman** back to talk some more about multimedia, particularly "Incorporating graphical content into multimedia presentations." Multimedia is, of course, one area where object technology excels. But multimedia is also probably an area that would have a difficult time being cost effective without objects, at least on the software side. (Another question for Ken and Louis might be "can there be widespread implementation of multimedia applications without objects?") There are other considerations, however, when looking into multimedia applications. Equipment, in particular, is one. I say equipment and not hardware because though multimedia may require what we traditionally recognize as computer hardware, i.e., touch screens, it also can include much more sophisticated sound and video equipment. I don't consider this to be a problem, just an expansion of our perspective. This expansion of perspective is a necessary element of consideration when planning to implement multimedia applications. I like the idea of closing the gaps between technological areas for us regular people. The arts and sciences (radio, television, film, medicine, chemistry, biology, etc.) have done extensive work in incorporating and utilizing sophisticated technologies to support understanding of their areas of study. It is now our turn to take advantage of alternative dimensions of our perceptive skills (tactile, visual, auditory) to enhance our comprehension. (I think I'm about to step on virtual reality. Where are my data gloves?)

Editor
Marie A. Lenzi

SIGS Publications, Inc. Advisory Board
Tom Atwood, *Object Design*
Grady Booch, *Rational*
George Bosworth, *Digitalk*
Brad Cox, *Information Age Consulting*
Chuck Duff, *The Whitewater Group*
Adele Goldberg, *ParcPlace Systems*
R. Jordan Kreindler, *General Electric*
Meilir Page-Jones, *Wayland Systems*
Tom Love, *Consultant*
Bertrand Meyer, *Interactive Software Engineering*
Sesha Pratap, *CenterLine Software*
P. Michael Seashols, *Versant Object Technology*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

Hotline Editorial Board
Jim Anderson, *Digitalk, Inc.*
Larry Constantine, *Consultant*
Mary E.S. Loomis, *Versant Object Technology*
Reed Phillips, *Knowledge Systems Corp.*
Trygve Reenskaug, *Taskon A/S*
Zack Urlocker, *Borland International*
Steven Weiss, *Wayland Systems*

SIGS Publications, Inc.
Richard P. Friedman, Founder & Group Publisher

Art/Production
Kristina Joukhadar, Managing Editor
Susan Culligan, Creative Director
Kristin R. Juba, Production Editor
Caren Polner, Desktop Designer

Circulation
Diane Badway, Circulation Business Manager
Kathleen Canning, Fulfillment Manager
John Schreiber, Circulation Assistant

Administration
David Chatterpaul, Bookkeeper
Suzanne W. Dinnerstein, Conference Manager
Laura Taylor, Conference Coordinator
Jennifer Fischer, Assistant to the Publisher
Jennifer Englander, Administrative Assistant
Margherita R. Monck, General Manager

Jane M. Grau, Contributing Editor

hotline on OBJECT-ORIENTED *technology*

The Hotline on Object-Oriented Technology (ISSN #1044-4319) is published monthly by SIGS Publications, Inc., 588 Broadway, NY, NY 10012, (212)274-0640. © Copyright 1991 SIGS Publications, Inc. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the U.S. Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher. Mailed First Class. Subscription rate — one year (12 issues) $249, Foreign and Canada $279. Single copy $25.

POSTMASTER: Send address changes & subscription orders to The Hotline, Subscriber Services, P.O. Box 3000, Dept HOT, Denville, NJ 07834.

Submit editorial correspondence to Marie A. Lenzi, 155 East 31st Street, #22S, New York, NY 10016. voice: (212)213-8701, fax: (212)725-0224

SIGS PUBLICATIONS

Publishers of *Hotline on Object-Oriented Technology, Journal of Object-Oriented Programming, Object Magazine, The X Journal, The C++ Report, The Smalltalk Report,* and *The International OOP Directory.*

Back issues available

Vol.3, No. 1/November '91 ≡ Leading the U.S. semiconductor manufacturing industry toward an object-oriented technology standard • *Hollowell* ≡ Coping with complexity: OOPS and the economists' critique of central planning • *Lavoie, Baetjer, & Tulloh* ≡ Choosing Object Technology: What's the object? • Rosenthal ≡ OOP: the MISsing link • *Rovira*

Vol.2, No. 12/October '91 ≡ A modest survey of OOD approaches • *Bulman* ≡ What is a "certified" object programmer? • *Bellm* ≡ Perspective: investing in objects today •Bowles ≡ Object oriented in Melbourne, Australia • *Haebich* ≡ The Object Management Group • *Gutman & Matthews*

Vol.2, No. 11/September '91 ≡ From applications to frameworks • *Urlocker* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Getting started with object technology: efffectively planning for change • *Lorenz* ≡ Object statistics on the way • *Lenzi* ≡ On objects and bullets • *Onart*

Vol.2, No. 10/August '91 ≡ Distributed object management: improving worker productivity• *Osher* ≡ Getting the best from objects: the experience of HP • *Coleman & Hayes* ≡ Applications: EC employs object technology ... • *Stein* ≡ Capacity Planning: Fiddling while ROMs burn • *Rovira*

Vol.2, No. 9/July '91 ≡ Multimedia is everywhere! • *Weiss & Berman* ≡ Developing an object technology prototype • *Kulmek* ≡ Object-oriented capacity planning • *Rovira* ≡ How OOP has changed our developmental lifecycle • *Zeik* ≡ Modularization of the computer system • *Argenius*

Vol.2, No. 8/June '91 ≡ Domain of objects: the Object Request Broker • *Dyson* ≡ Object-based approach to user documentation • *Reeder* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Do we need object-oriented design metrics? • *Hopkins*

Vol.2, No.7/May '91 ≡ Hybrid object-oriented/functional decomposition for software engineering • *Henderson-Sellers* ≡ So, what makes object databases different? (Part 4) • *Blakey* ≡ Using the generic application to solve similar domain problems • *Gossain* ≡ Experiences using CLOS • *Hopkins* ≡ International Conference on Object-Oriented Technology, Singapore • *Bennett*

Vol.2, No.6/Apr. '91 ≡ An artist's perspective of programming • *Rovira* ≡ So, what makes object databases different? (Part 3) • *Blakey* ≡ Moving from Pascal to C++, Part 3 • *Gole* ≡ Object projects: what can go wrong • *Stewart* ≡ Reflections from LOOK-'91 • *Anders-Angenius*

Vol.2, No.5/Mar. '91 ≡ Large-scale object systems development • *Morgan & Davis* ≡ Data definition languages • *Blakey* ≡ Object disorientation • *Kerr* ≡ Moving from Pascal to C++, Part 2 • *Keller* ≡ The Object Management Group • *Barber*

Vol.2, No.4/Feb. '91 ≡ Moving from Pascal to C++ • *Sollisch* ≡ The Australian object-oriented scene • *Henderson Sellers* ≡ So, what makes object databases different? • *Blakey* ≡ Managing the evolution • *Novobilski* ≡ 1990: the year of realism • *Love* ≡ Choosing a configuration for object development • *Moody*

Vol.2, No.3/Jan. '91 Special issue: CASE and Methodology ≡ Object orientation: stop, look and listen! • *Page-Jones* ≡ The spiral model for object software development • *Wasserman & Pircher* ≡ Bootstrapping object-oriented CASE • *Wybolt* ≡ A CASE tool for object software development • *Ackroyd* ≡ Comments on "Race for the database pennant" • *Loomis*

Vol.2, No.2/Dec. '90 ≡ Waiting for OODB2 • *Durham* ≡ Object modeling • *Singh* ≡ Object-oriented COBOL — the next generation • *Adams & Lenkov* ≡ Object-oriented training in the UK • *Hopkins*

Vol.2, No.1/Nov. '90 ≡ Objects by teamwork • *Constantine* ≡ Race for the database pennant: a third-generation manifesto • *Stein* ≡ Project team descriptions *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.12/Oct. '90 ≡ Crown jewels: the value of OOT in MIS departments • *Bonar* ≡ Why is an object database faster than a relational system? • *Loomis* ≡ Visual programming object language environments • *Klausner* ≡ Hype — necessarily (or a necessary) evil? • *Adams* ≡ On Managing: Déjà Vu • *Lenzi*

Vol.1, No.11/Sep. '90 ≡ Introducing OOPS through design — it's all in the words • *Copen Goldstein* ≡ The Australian object-oriented scene • *Henderson-Sellers* ≡ Objects on tour (Review of Borland's OOP World Tour) • *Lenzi* ≡ Design principles for modular product development • *Bennett* ≡ Asset management and object-oriented technology • *Hopkins & Warboys*

Vol.1, No.10/Aug. '90 ≡ WyCASH+: An application built within an OOP environment • *McCullough & Deshler* ≡ Mathematics and methodology • *Martin* ≡ Organizing for reuse • *Goldberg & Rubin* ≡ CALS Expo–89 Conference • *Purdy*

Vol.1, No.9/July '90 ≡ Cognos and Eiffel: a cautionary tale • *Leathers* ≡ What's in an object anyway? • *Novobilski* ≡ The impact of Windows 3.0 on the OOP market • *Urlocker* ≡ Introducing reuse into the development process • *Goldberg & Rubin*

Vol.1, No.8/June '90 ≡ A taste of the USENIX C++ Conference • *Wells* ≡ Comparing relational and object database management systems part 2 — Composites, flexibility, and extensibility • *Martin* ≡ Case studies in prototyping, part 3 • *Goldberg & Rubin* ≡ Object-oriented requirements analysis • *Berard*

Vol.1, No.7/May '90 ≡ Object-oriented development: not business as usual • *Lorenz* ≡ Comparing relational and object database management systems part 1 — Single paradigm programming is the key difference • *Martin* ≡ Case studies in prototyping, part 2 • *Goldberg & Rubin* ≡ Understanding the recursive/parallel lifecycle • *Berard*

All back issues of *Hotline on Object-Oriented Technology* are available. Please call (212)274-0640 for details.

To order, see back page; for reprints of individual articles, call Michael Biggerstaff at Reprint Management Services (717) 560-2001.

Calendar

The Hotline Calendar presents conferences and meetings that focus exclusively on object-oriented technology. To have a meeting or conference listed, please send the dates, conference name and location, sponsor(s) and contact name and telephone number to the Editor: Marie A. Lenzi, 133 East 31st Street, #22S, New York, NY 10016, fax (212) 725-0224.

February 3–5, 1992 **LOOK '92** Copenhagen, Denmark Sponsor: Niveau, *Object Magazine*, & *The Journal of Object-Oriented Programming* Contact: Miles Silverman (212) 274-0640	**February 5–7, 1992** **OOP '92** Munich, Germany Sponsor: *The Journal of Object-Oriented Programming & Object Magazine* Contact: Miles Silverman (212) 274-0640	**February 11–21, 1992** **C++ Across America** SF, LA, Dallas, Chic., Bos., NYC, Wash. D.C. Presented by: *The C++ Report* Sponsored by: Microsoft Contact: (212)274-9135	**February 25–28, 1992** **Software Development '92** Santa Clara, CA Sponsor: Miller Freeman Contact: (415)905-2414	**March 23–26, 1992** **DB/Expo '92** San Francisco, CA Sponsor: NDN Contact: (415)941-8440

New Partnerings and Acquisitions, December 1991, continued

Hewlett-Packard Corp. has agreed to adopt **Microsoft Corp.'s** object technology in its NewWave graphical computing environment. Under the agreement, HP will support Microsoft's Object Linking and Embedding (OLE) technology in NewWave, making it possible for OLE applications to run under NewWave. Also, HP will join an industry group advising Microsoft on the object extensions to OLE due in the next release.

Instantiations has been elected as a member of **Sequent Computer System's** new Object Technology Partnership program. Under the partnership, Instantiations will release its Convergence/Team Engineering Environment on the Sequent Symmetry 2000 multiprocessing system. Instantiations will also provide object-oriented consulting and educational services to Sequent customers.

ONTOS, Inc., a vendor of object-oriented database systems, announced **IBM** has selected the company as an Industry Application Specialist (IAS) for the multimedia market. According to IBM, ONTOS was selected as an IAS in multimedia because it offers a high-quality object database, combined with proven industry and product expertise, and quality customer marketing and support program. Earlier this year, ONTOS joined the IBM Business Partner program as the first object database vendor.

John Diebold, chairman, **The Diebold Company, Inc.,** will chair the first ever conference and expo on multimedia to be held October 12-16, 1992, at McCormick Place in Chicago. Well known as a "futurist," Diebold is an internationally acknowledged leader in the fields of management and technology. As a pioneer in the field of automation, he is recognized as the person who coined the word "automation" in its present meaning and had much to do with introducing it to general usage. Diebold will also chair a Multimedia World show series planned to take place following the Chicago event.

ExperTelligence has announced integration of ACTION! design and development tools with **ITASCA's** object database management system. ITASCA is a provider of object database management systems and a member of the MCC Consortium. ExperTelligence provides object-oriented interface and application development systems for microcomputer and workstations.

Versant Object Technology Corporation has announced that **IBM** Austria has selected the VERSANT Object Database Management System for marketing and distribution by IBM in Austria. Under the new distribution agreement, IBM will sell and support the VERSANT ODBMS running under AIX on the IBM RS/6000 as a recommended software solution to IBM customers developing applications in a number of areas, including concurrent engineering, computer-integrated manufacturing, computer-aided design (CAD), computer-aided software engineering (CASE), document/image management, and office automation.

Versant Object Technology Corporation has announced an agreement with **GE Corporation** Research and Development to obtain marketing and technology rights to GE's OM Tool, a graphical, UNIX-based software engineering tool that automates key aspects of the object modeling technique developed at GE over the past five years. Under the agreement, Versant will commercialize the OM Tool technology as a stand-alone product called the VERSANT Object Modeler. GE and Versant will also participate in ongoing technology exchange, so that extensions to the implementation by GE of the object modeling technique will be provided to Versant for incorporation into future commercial releases of the VERSANT Object Modeler.

Versant Object Technology Corporation and **Rational**, of Santa Clara, CA announced that Rational has selected VERSANT as the object database management system to be incorporated into Rational Rose, Rational's recently announced graphical software engineering tool. Rational will distribute the VERSANT ODBMS as a runtime version included in Rose, initially on UNIX workstations.

The founding members of the **Advanced Computing Environment** (ACE) initiative and **UNIX System Laboratories** (USL), a subsidiary of **AT&T** have announced agreement on a common set of application programming interfaces for the ACE UNIX system environment, one of the two operation system environments supported by the ACE initiative. This unified open platform definition will enable systems based on the Open Software Foundation's OSF/1 operating system or USL's UNIX System V, Release 4 (SVR4) to provide a common set of source-compatible application interfaces, allowing developers to develop applications and be assured that they will run on all ACE UNIX system platforms. This set of UNIX system interfaces will be available for both ACE hardware platforms — Advanced RISC Computing (ARC)-based and x86 systems.

USL has announced that it is joining the ACE initiative and will ensure that its ACE SVR4-based product family supports the same set of interfaces that include the Open Software Foundations' (OSF) Application Environment Specification (AES) for the operating system, Motif graphical user interface, and Distributed Computing Environment (DCE).

Compaq Computer Corporation and **USL**, in cooperation with **The Santa Cruz Operation, Inc.** (SCO), are working to propose to the ACE initiative UNIX vendors a common application interface specification for a desktop manager.

... continued from page 1

Now we know, from a high-level perspective, that an object is really a representation of the natural way to view the world. It's taking into consideration the what and how of any entity and being able to easily represent such an entity in pieces of computer code. The abstraction mechanisms so natural to human beings can now be effectivity utilized in developing computer systems. There is a natural structure to objects. We can provide templates for such entities so the constructs are always known. We call these templates types. We also see a certain "naturalness" in the way one would build relationships between types, providing us with a feature known as subtyping. Inheriting functions and characteristics between these types seems a logical thing to do as a next step. Basically, what I discovered is that when I threw out the things I had been forced to learn to work with computers and let my natural logic work for me, objects became easy to understand.

THE BENEFITS OF OBJECT TECHNOLOGY

The benefits started to become obvious as well. Anyone requiring a complex application would benefit from ways used to simplify the task — to ideally map thoughts and ideas directly onto pieces of code. The ease of the structuring through typing and the logic of inheritance led me to understand how easy it might be to make changes to the code — everything would be modular in the system. The faster one can make quality changes, the sooner a system could be extended or a new system built. These are the benefits that make sense to a customer in any industry. These are benefits that Digital found to be valuable in building our own software products.

I was surprised to eventually discover pockets of object-oriented engineering activity in a number of software and hardware groups where traditional programming methodologies were the norm. There were manufacturing applications being built. There was object work being done for our dictionary development, in our CAD systems, and in the development and extension of languages. Almost all of this activity was self-motivated since there wasn't any formal strategy available at that time. What was available in some of our software engineering facilities were workstations with object-oriented application development environments installed for anyone to use. This provided a nurturing, nonaggressive learning environment for developers to sit and discover. I highly recommend creating such an environment for those in your MIS shops to work with. Find your best and most open developers and encourage their interest in objects. The effect is contagious between engineers. Shortly thereafter I was finding a large number of object proponents at Digital.

Once I felt that I had much of the data on the object model, it was time to begin presenting the benefits and business justifications for developing object-oriented products and promoting object-oriented design. When it was thought that just one or two presentations would be all that was required, we found that those of us chartered to present had to do so, to the same people, many times. It didn't matter necessarily how clear the information was because, as mentioned, people need to put aside their current view of computing before the object paradigm can be really understood.

> ***"When I threw out the things I had been forced to learn to work with computers and let my natural logic work for me, objects became easy to understand."***

In the Digital situation, it was important to decide how deeply we would move into the technology since our customers would be affected. Basically, the idea was to take small steps, but important ones. We were selective in choosing projects. As products evolved and our business partners began working with us on delivering object technology, there were many reviews by engineering and management to ensure that the customer and Digital processes were affected as little as possible. Training for customers and internal users was put together quickly and effectively to aid in education. Training is critical to those using this methodology.

In very little time (less than one and one-half years), Digital felt that committing to object technology was critical. Still, the rollout of it had to be monitored and supported. An object-oriented program office was developed to optimize Digital's experience in the technology and to provide a cohesive object-oriented architecture for object-based and traditional products. It was paramount to ensure that existing technology could work well with the new. The program office was staffed primarily by a software architect and program manager. A number of key individuals acted as consultants to the program office on a regular basis.

Currently, we are in a very exciting period with object technology. The question now posed internally is not "What are objects?" but "How do we best deliver the technology as solutions for the customer?" I believe in time we will move from promoting the fact that we have object technology to promoting the idea that the technology has merely optimized application and tool development. I believe that much of the core Digital product development will focus on the object paradigm. It will become simply an enabling technology for bet-

ter product and project development. It will become a given in the industry rather than something that must be uniquely called out. It is that valuable — even in its current evolutionary stage. Of course, in addition to object-based applications, DEC will continue to offer object-oriented languages, tools, and databases so our customers will have the base functionality to do their own effective application development.

For software vendors, object technology has finally been accepted as being important to development activities. There are a few vendors, because of their current fiscal situation, lack of knowledge, or investment policy, that choose to ignore the technology. I believe that this is unfortunate since those who commit to the technology will have the competitive edge over time. I think the same will hold true for the MIS shops of the future. Object qualities will become vital for key applications, even if it's simply in the form of better designed graphical user interfaces. Not every application will require pure object implementations, but the benefits of the paradigm will frequently make it desirable.

MOVING TO OBJECT TECHNOLOGY

I think the evolution that I've gone through in DEC, which includes a period of "object fear" is applicable to almost anyone. There are a number of points to remember when you begin your adaptation. First, something that wasn't obvious to me when I started my investigations is that object technology is here, it is real, it is beneficial, and it will grow in prominence. If you can accept that, the next steps will help you as a manager involved in system or application development:

- Walk into your education with an open mind and open eyes. You will need training in the paradigm, and training is best delivered by consultants well versed in the technology. Review their offerings to make sure that object education is a major piece of their portfolio, not merely a minor response to a vital trend. When possible, the training should be presented in a general fashion, not tied to any specific product. You need a clear field of vision since there is a lot of hype in the industry today. What you don't need is a product sales job at this point.
- Ask questions. If you don't understand a concept, force the educator to make it clearer … you won't be alone. I have found that many professionals feel an internal pressure to understand the concepts quickly. You will be required to think "differently" on the object model, so give yourself some leeway in adapting.
- Read. There are a number of object journals available today, and articles on object technology are showing up in all types of software publications. There are also a number of consultant studies available that can be very beneficial.

Once you get to the point of understanding and now want to do a knowledge transfer and maybe prototype a project to test feasibility, try the next steps:

- Pick the right developers and keep the project small. In any MIS shop of reasonable size, there should be at least one or two individuals interested in the technology. Educate them the way you've educated yourself. Allow them to get excited about the possibilities. Don't force feed it.
- Pick the right tools. For some, the extended languages like C++ are just right; for others, products like Trellis or Smalltalk would be better. Make sure the developers get product training once the tool is selected.
- Start from the language perspective. Even if you know you'll need a database eventually, keep the project small and get comfortable with a language. Learning the rest, such as an object database, will be much easier if you master the language first. Object-oriented design and analysis is evolving and will help with the adaptation over time.
- Assume you will toss much, if not all, of what you've prototyped out the window. The key to good object-oriented programs is in the type hierarchies. It takes time and attention to detail to get these right, and few get it right the first time.
- Allocate sufficient time for education and development. There is a learning curve to the data model and tools.
- Understand the realities. It is not in the initial application development where many of the positive points of object orientation exist. It's in the aftermath, when enhancements and maintenance are an issue. Here is where you will see the cost effectiveness.
- Don't underestimate the power of data abstraction. We've discovered that only because of the powerful data abstraction capabilities in object languages were some applications feasible. If you work in a complex environment, this point alone can justify your quick move to object technology.

As a last point, dare to be the visionary and expect some negative responses. Many people are afraid of change and would rather see current ways of doing business continue regardless of the ramifications. Your goals to build better software and keep costs under control can be realized with your move to object technology. ≡≡

Lydia Bennett has been with Digital Equipment for eleven years and has focused on new product development. She is currently a Strategic Marketing Manager for Emerging Technologies, and her specialty is object technology.

Convergence/Team Engineering Environment

Instantiations, Inc.
921 SW Washington, Ste.312
Portland, OR 97205
(503) 242-0725

Instantiations, Inc. has announced **Convergence/Team Engineering Environment**, a multirepository team programming environment for Smalltalk. Convergence/Team enables programming teams to effectively develop commercial and industrial applications using ParcPlace Systems' Objectworks\Smalltalk. Convergence Team's shared repository environment allows team members on different types of computers to simultaneously create, browse, access, and share code. Size and number of repositories is limited only by available disk space. Instantiations also announced Convergence/Team Services. Consulting, training, mentoring, design and prototyping support and custom engineering are available. Further, the company announced that it will produce a Convergence-compatible version of its well-known Application Organizer Plus, a smaller scale Smalltalk development environment.

MDBS IV and M/4 Windows-based language interfaces Object/1 Version 2.1

Micro Data Base Systems, Inc.
Two Executive Dr.
P.O. Box 6089
Lafayette, IN 47903-6089
(317) 463-2581

Micro Data Base Systems, Inc. announced three new **Windows-based language interfaces for MDBS IV and M/4 for Windows**. The new language interfaces include Microsoft Visual Basic, Borland C++, and Zortech C++. These new language interfaces allow rapid application development of robust database applications and added flexibility for application developers. MDBS IV is a high-performance online transaction processing DBMS that utilizes entity-relationship modeling and client/server architecture. Ultra-large databases and mission critical applications that demand performance are the most likely candidates for MBDS IV usage. Other MDBS IV language interfaces include C, C++, Pascal, COBOL, Fortran, BASIC, KnowledgeMan, GURU, MPE/XL, and VAX VMS. M/4 for Windows is a stand-alone Windows DBMS.

Micro Data Base Systems has also released **Object/1 Version 2.1** for the Windows 3.0/3.1 graphical environment. This release of Object/1 adds bitmap support, improved font handling, and a significant performance increase to the Forms Painter and TBL Designer. An Object/1 Tutorial will be shipped with each update to aid developers of Object/1 Windows applications. Object/1 is an object-oriented graphical development environment for the creation of Microsoft Windows 3.0/3.1 and OS/2 PM applications. Object/1 comes complete with a relational database system, TBL, and can interface with MDBS IV, SQL Server, Sybase, and IBM Database Manager utilizing the object classes within the Object/1 Professional Packs.

ANSI C and C++ tools

Microtec Research Inc.
2350 Mission College Blvd.
Santa Clara, CA 95054
(408) 880-1300
(800) 950-5554

Microtec Research Inc. has announced the availability of versions of its **ANSI C and C++** tools that provide compatibility with VxWorks from Wind River Systems, Inc. Users of the VxWorks real-time operating system will now be able to build, download, and debug applications using the industry standard Microtec Research optimizing compilers for Motorola 680x0 family microprocessors. The combination of the Microtec Research and Wind River systems tools allows the software engineer to build C and C++ applications using the Microtec Research MCC68K and CCC68K compilers, link with the Wind River Systems real-time interface libraries, and download the resulting module into the VxWorks environment. The Microtec Research tools also benefit users of Wind River Systems' VxGDB source-level debugger by supporting VxGDB load module file formats.

New Partnerings and Acquisitions, December 1991

Digital Equipment Corporation has announced that **AICorp**, Inc. has joined its Complementary Marketing Program. Under the CMP agreement, Digital and AICorp sales representative jointly market and promote KBMS, AICorp's premiere application development tool, and Digital's VAX computing systems throughout the United States and Canada.

ONTOS, Inc. has announced its support for **IBM**'s OS/2 2.0, an advanced 32-bit operating system. ONTOS was the first object database company selected as an IBM Business Partner and is an IBM Industry Application Specialist for multimedia.

Synthesis Computer Technologies Inc. has announced a joint marketing agreement with **NCR**. The agreement pairs Synthesis' **Unix** software development tools, CASE/AP, with NCR's System 3000 for sales to the corporate, educational, and government marketplaces.

Data General Corporation and **Digital Analysis Corporation** have signed a joint agreement to provide public sector and commercial customers with network management capabilities over both TCP/IP and OSI communications networks. With this agreement, the two companies will design and develop an integrated SNMP-CMIS/CMIP network manager.

IBM and **Sapiens International Corp.** are teaming up to jointly market SAPIENS, an application development and maintenance product that uses rule-based, object-oriented technology to build and execute data processing applications on IBM mainframes.

PRC Inc. is forming an independent subsidiary to distribute leading-edge open systems product and technology to commercial and federal integration markets. The group's first effort will be to fulfill the terms of a strategic partnership signed recently with **Object Design Inc.** PRC will resell ObjectStore, while Object Design will provide support services.

VERSANT Object Modeler

Versant Object Technology Corp.
4500 Bohannon Dr.
Menlo Park, CA 94025
(415) 325-2300

Versant Object Technology Corporation has formally announced the **VERSANT Object Modeler**, a powerful, highly interactive, graphical data modeling tool that simplifies the task of designing and creating object-oriented applications. The VERSANT Object Modeler allows object designers and programmers to specify class hierarchies through simple pointing and clicking operations. The resulting object model is maintained in a storage-independent form. Code is then generated to automatically create an object or relational database in a number of possible forms, including C++ classes, C structures, and SQL statements. Availability on the IBM RS/6000 and on other UNIX platforms will follow.

Objectivity/DB integrated graphical environment

Objectivity Inc.
800 El Camino Real
Menlo Park, CA 94025
(415) 688-8000

Objectivity Inc. has announced a fully integrated graphical environment for developing object-oriented databases and applications based on SoftBench and C++Bench from Hewlett-Packard Company. The integration with HP's SoftBench and C++/SoftBench provides **Objectivity/DB** users with an object-oriented database development environment that integrates SoftBench and C++/SoftBench software development tools with Objectivity/DB database application development tools. Intelligent links between Objectivity/DB, SoftBench, and C++/SoftBench enable developers to browse the data and schema definitions stored in the database and view the classes and methods that control the data. Objectivity has also enhanced HP's C++ Developer and other C++/SoftBench tools to recognize Objectivity's Data Definition Language (DDL). This allows an Objectivity/DB database developer to use HP's C++ Developer to graphically design Objectivity/DB databases, define classes and methods, check syntax, and automate other Objectivity/DB tasks.

AppEvent Object

Serius Corp.
1981 E. 4800 South
Salt Lake City, UT 84117
(801) 272-7788
(800) 876-6847

Serius Corp. has introduced the Serius **AppEvent Object**, which allows applications built with the Serius Programmer and Serius Developer desktop programming environments to communicate with other Macintosh applications supporting Apple Events. Users of the Serius AppEvent Object don't have to understand how Apple Events are implemented at the code level to program with them. The AppEvent Object can send and receive Core Apple Events. AppEvent incorporated facilities that allow users to define their own Apple Events and to subscribe to existing and future Apple Events defined by other organizations. Serius-built applications can also send and receive arbitrary Serius Objects using AppEvent.

Smalltalk\SQL platform support

Synergistic Solutions Inc.
63 Joyner Dr.
Lawrenceville, NJ 08648
(908) 855-7634
(609) 586-0025

Synergistics Solutions Inc. has announced additional platform support for **Smalltalk\SQL**, the portable database interface for Smalltalk. The product works in conjunction with ParcPlace Systems' Objectworks\Smalltalk to enable graphical user interface (GUI) applications to access information stored in relational databases. Direct database support is currently available for the Sybase and Oracle databases. DB2, Informix, Ingres, Rdb, and other databases may be accessed through a variety of gateway products. Several other direct interfaces will be announced later this year. Smalltalk\SQL is a set of portable classes that encapsulate database access through an object-oriented metaphor called an SQL agent. Developers simply "plug in" the appropriate agent for their target database. There are no modifications necessary to applications using ANSI SQL. Pluggable SQL agents are capable of instantiating objects based on arbitrary queries including relational joins. Smalltalk\SQL includes a portable SQL agent class, several browsers and development tools.

InfoWare Version 1.0

Ensemble Software Systems, Inc.
555 Bryant St., Ste. 347
Palo Alto, CA 94301
(415) 325-2773

Ensemble Software Systems, Inc. has announced **InfoWare Version 1.0**, a database connectivity package that provides an object-oriented interface to relational databases for applications written in ParcPlace Systems' Objectworks\Smalltalk. Applications built with InfoWare are able to access relational data in the form of objects. The applications are written entirely in Smalltalk, instead of the embedded SQL of the host server, providing the developer with a uniform object-oriented interface and portability across a variety of RDBMS servers. InfoWare 1.0 is designed for database application developers using Objectworks\Smalltalk Release 4. InfoWare includes an application program interface that presents relational data to applications in the form of persistent objects. InfoWare also includes design tools that allow developers to easily extract object definitions from existing database tables, create new tables from existing object definitions, and keep tables and object definitions mutually consistent across design changes. A runtime tool set allows users to retrieve and manipulate persistent objects graphically and may be used stand alone or integrated into other applications.

ExperClub

ExperTelligence, Inc.
5638 Hollister Ave., 3rd Fl.
Goleta, CA 93117
(805) 967-1797

ExperTelligence, Inc. has established a new user organization, **ExperClub**, for object-oriented programming professionals who use the company's Action! intelligent interactive interface builder with one or more of the OOP languages supported by Action!. ExperClub is organized around OOP language communities, with the first section comprised of users of Apple's Macintosh Common Lisp 2.0 and other sections to be added. Members have unlimited access to Action/MCL programmers and developers by mail, facsimile, or telephone. A program of seminars and club-sponsored meetings is in development. An advanced object-oriented programming technology newsletter, *ExperTechNotes*, focuses on questions, problems, and solutions submitted by OOP developers.

GETTING STARTED

Adopting objects: a path

by J. Michael Connell

Corporate leaders in America have seen what damage can befall a firm when it lags behind the competition in adopting critical computing capability.

These same leaders have also seen the harm that can come from being too hasty to commit critical programs to a new technology before it is ready. To move too slowly in adopting a new technology can be dangerous, but to move too quickly is equally dangerous. Object technology is not an exception. Any object technology adoption program should be undertaken with equal parts of enthusiasm and caution. A well planned and executed process will mediate many of the risks and ensure the best timing.

It is estimated that as many as one-third of the major corporations in the U.S. are investigating objects to some degree or another. For most of these companies, this process will be unstructured, ad hoc, and random. Such a process may take the form of a widely dispersed and loosely associated collection of projects. Each of these projects will have its own parochial goals. And, over time, this process will lead to a competent understanding and trust of objects. A more focused and purposeful approach to evaluating and adopting objects can be quicker and less costly.

PHASED TECHNOLOGY ADOPTION

Is there a best way to go about validating object technology? Every company seeking to adopt new technologies goes about it in a somewhat different manner. This is also true for objects. The lore surrounding objects is filled with stories of difficulties and triumphs. It can sometimes be arduous to identify what's really happening from this profusion of anecdotal evidence; however, there are enough similarities that a generalized model of the process can be formed. These similarities form the basis of a framework or model for the technology adoption process.

A typical company's process for adopting objects will include three phases, where each phase is divided into two stages. These are:

I. Exploration phase
  - Stage 1. Prospecting
  - Stage 2. Proof-of-concept

II. Technology transfer phase
  - Stage 3. Evaluate alternatives
  - Stage 4. Pilot trial

III. Deployment phase
  - Stage 5. Initial production
  - Stage 6. Diffusion

This model can be applied to large and small companies alike. A large company will typically follow this or a similar process step by step. However, small companies often employ only pieces of the process because they are generally time sensitive and willing to take greater risks.

A prudent company will proceed in an orderly fashion to balance investment and risk. When little is known about a new technology, like objects, it must be considered high risk; therefore, a company should be wary about making large commitments of money and effort. A small investment is the wiser course. Such an investment will foster a deeper understanding of the value of objects and, if successful, will lead toward increasing larger investments. This policy of risk management by creeping commitment is practiced by most major corporations.

J. Michael Connell is a consultant in Portland, OR specializing in the marketing and management of object-oriented products, tools, and environments. During most of the formative years of object-oriented systems, he has been in roles key to the development, market positioning, and penetration of this technology.

EXPLORATION PHASE

The first phase of the technology adoption process is the search for likely candidate technologies. Large companies typically have one or more departments whose sole purpose is to track and evaluate the development of new computer technologies. Let's refer to this function as the "gatekeeper lab." Gatekeeper labs locate, study, characterize, and monitor any and all new technologies that show a potential of becoming important to the company. The activities of these groups form stage 1, prospecting, and stage 2, proof-of-concept, of the technology adoption model.

Stage 1. Prospecting

The purpose of this stage is to acquire sufficient knowledge about objects to separate the truth from hype. They must answer the following questions: Are objects real? How could this technology be important to my firm? To do this, the staff of the gatekeeper lab will track the technical literature; attend conferences, workshops, and short courses; and speak with innovators in the technical and academic communities. After they are satisfied that objects are a promising new technology, a more thorough technology analysis study will be conducted.

To a great extent, the process of selecting objects should be driven by the computing needs of the corporation. However, it is often the case that the benefits of an emerging technology may not be apparent from the onset. Therefore, it may well be wise to evaluate objects even though the tie to the corporation's needs may be speculative at first.

Stage 2. Proof-of-concept

After having determined that objects appear to be real and may have some relevance to the needs of the corporation, the gatekeeper lab will begin the proof-of-concept stage. The purpose of this stage is to determine if the objects truly do deliver on the promises of their proponents. Ideally, this project would also demonstrate the value of the objects in the context of the company's business.

The initial proof-of-concept project is typically a company's first attempt to use objects in a meaningful way. This project should be a prototype effort only, with no consideration to using the resulting output for any practical purpose beyond evaluating the technology. Several proof-of-concept projects are often required for a rigorous evaluation. These projects should be structured in such a manner that they reflect the way the company might use objects in its business applications. A successful completion of this stage should validate this emerging technology as a viable candidate for use in a production environment.

TECHNOLOGY TRANSFER PHASE

With the successful completion of the exploration phase, the technology of objects is now ready to be transferred from the gatekeeper lab into a production software development unit. This phase has three purposes: to evaluate objects more thoroughly in the light of the company's computing needs; to transfer the knowledge, expertise, and know-how of objects from the gatekeeper lab to a core group of application developers; and to use objects in a medium-scale project to evaluate this technology in a production environment.

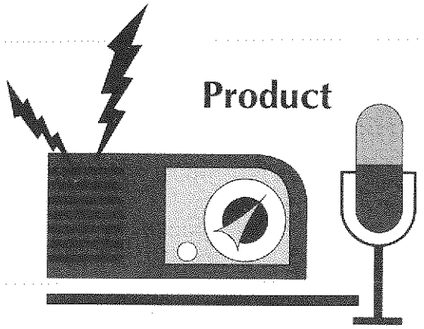
The technology transfer phase has two stages: evaluate alternatives and pilot trial. These stages need not be sequential; however, it is generally better to evaluate alternative technologies and select one or two before proceeding with pilot trials. This sequence will tend to minimize the investment in the technology transfer phase. Sometimes though, it may be necessary to proceed with one or more pilot trials before sufficient information is available to make a selection.

Stage 3. Evaluate alternatives

At this point, the focus of the technology adoption process switches from looking at the upside benefits of objects to identifying and evaluating the potential downside risks of their adoption. Evaluators must ask the tough questions: How will this affect my organization and my software engineering process? How do I develop and maintain a library of reusable software components? How will it affect my testing procedures? How will I do deployment, maintenance, and repair? And the most important questions are: What problems will objects solve for me? What other ways can I achieve similar results? What are the impacts of these alternatives? What are all the costs involved in switching over?

Experience shows that the move to objects is rarely an isolated event. In most cases, companies are attempting to adopt several new technologies at the same time, and objects are only one of them. The most common forcing function is the desire to move to a more distributed computing environment. Corporate managers, when attempting this change, are finding that many of the well-established software engineering methodologies and techniques that were entirely adequate in the centralized computing case do not address all of the issues encountered while developing and maintaining applications in a distributed environment. Therefore, the task of evaluating objects is often part of a task to evaluate the broader combination of distributed computing technologies. The ease with which these technologies interplay with each other becomes mandatory criteria in this evaluation process.

The purpose of this evaluation stage is to weed out unacceptable alternative solutions before proceeding on



Product Announcements

Product Announcements is a service to the readers of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY; it is neither a recommendation nor an endorsement of any product discussed.

HP OpenODB

Hewlett-Packard Co.
19310 Pruneridge Ave.
Cupertino, CA 95014
(800) 752-0900

Hewlett-Packard Company has announced **HP OpenODB**, a commercial object-oriented database management system (ODBMS) for large, multiuser environments. HP OpenODB will be available for HP as well as non-HP platforms. HP OpenODB can be ported to other hardware vendors' platforms because of its object-oriented structured query language (OSQL). OSQL builds on current database technology by allowing programmers to use languages such as C, COBOL, Pascal, Fortran, and C++ to write applications. OSQL takes existing structured query language (SQL) features used in relational database technology and adds in the full power of objects. This allows HP OpenODB to offer the full functionality of robust relational database management systems such as HP ALLBASE/SQL.

ENVY/Developer

Object Technology International, Inc.
1785 Woodward Dr.
Ottawa, Ontario, Canada
K2C OP9
(613) 228-3535

Object Technology International, Inc. has announced the release of its high-productivity software engineering environment for ParcPlace's Objectworks\Smalltalk Release 4. The product, **ENVY/Developer**, provides a powerful concurrent software engineering environment for systems and applications development. Team support, version control, and configuration management are seamlessly integrated with Objectworks\Smalltalk's programming environment. ENVY/Developer is a multiuse tool set for delivering large systems incorporating advanced object-oriented technology. By maintaining all source and object code online in a shared repository, ENVY/Developer provides development teams with environment support for software reuse. The initial release will support five Objectworks\Smalltalk platforms: Windows 3.0 and Sun-3, Sun-4, RS/6000, and Apollo. Supported networks include UNIX TCP/IP, Novell Netware, and LAN Manager.

GeODE

Servio Corp.
1420 Harbor Bay Pkwy.
Alameda, CA 94501
(510) 758-6200

Servio Corporation has introduced the GemStone Object Database Development Environment, **GeODE**, a suite of development tools for visually and graphically designing and building ODBMS applications. GeODE was designed for applications developers whose experience is based on traditional programming methods and languages. GeODE provides a tightly integrated set of tools that allows developers to visually create screens, graphically direct application flow, and customize the application through graphical point-and-click interaction. Other GeODE tools include debuggers, cross-reference tools, and graphical browsers. The initial release of GeODE supports SUN SPARCstations running OSF's Motif and Sun Microsystem's OPEN LOOK. Phase two of GeODE will support Presentation Manager, Windows, and the Macintosh. Ultimately, GeODE will run on the complete line of operating systems and platforms supported by GemStone, including IBM, Digital, and Sun.

FACETS\4GL interface builder

ParcPlace Systems
1551 Plymouth St.
Mountain View, CA 94043
(415) 691-6700

ParcPlace Systems has announced that the **FACETS\4GL** fourth generation language (4GL) application development tool for Objectworks now includes an interface builder. FACETS\4GL is developed by Reusable Solutions and marketed by ParcPlace Systems. With the FACETS interface builder, applications can be enhanced easily with different kinds of buttons. An event-driven architecture ensures that actions and scripts are attached to the buttons. In addition, color selection has been simplified so users can point and click to select from a color palette. Users can also change the default styles to alter the look and feel of an entire application in one step. FACETS is a comprehensive suite of development tools that includes complete source code and an online hypertext-based context-sensitive help system. The system generates Smalltalk code, which can be incorporated into other Smalltalk applications. Applications developed with FACETS\4GL are completely portable across all ParcPlace supported platforms.

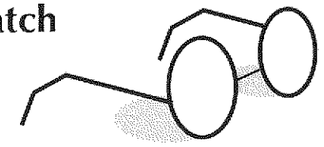
Tigre Programming Environment with Objectworks\Smalltalk

Tigre Object Systems, Inc.
3004 Mission St.
Santa Cruz, CA 95060
(408) 427-4900

Tigre Object Systems, Inc. now bundles the **Tigre Programming Environment** with ParcPlace's Objectworks\Smalltalk object-oriented language. The Tigre Programming Environment, which uses ObjectWorks\Smalltalk as its scripting language, lets developers create state-of-the-art graphical user interface (GUI) programs that run without porting on Macintosh II, Microsoft Windows 3.0, and all popular UNIX workstations.

Send Product Announcements to Marie A. Lenzi, 155 East 31st, #22S, New York, NY 10016.
Fax (212)725-0224. Include company name, address, and phone number.

Book Watch



The "Book Watch" column does not contain book reviews. These listings are abstacted from press releases provided by the publishers, and no endorsement is implied. Please send announcements to the Editor: Marie Lenzi, 225 Rector Place, Suite 12N, New York, NY 10280, fax (212) 786-4032.

Class Construction in C and C++: Object-Oriented Programming Fundamentals

Roger Sessions; Prentice Hall, Englewood Cliffs, NJ

A complete, in-depth introduction to object-oriented programming, developing classes, and C++ for C programmers.

Object-Oriented Programming with C++ and OSF/Motif

Douglas A. Young; Prentice Hall, Englewood Cliffs, NJ

Young takes the object-oriented approach to designing and building applications for interactive window-based user interfaces using C++ and the OSF/Motif user interface toolkit.

Modula-3

Samuel P. Harbison; Prentice Hall, Englewood Cliffs, NJ

Provides programmers with a clear and complete description of the new Modula-3 programming language and shows them how to write clear and correct code.

Object Lifecycles: Modeling the World in States

Stephen Mellor & Sally Shlaer; Prentice Hall, Englewood Cliffs, NJ

A companion volume to their *Object-Oriented Systems Analysis: Modeling the World in Data. Object Lifecycles* takes a look at object-oriented analysis based on building three types of fundamental models — information models, state models, and process models.

... FYI, continued from page 13

CASE

... Case technology deals with all aspects of the software automation process, not in each individual product, but within the totality of Case offerings. Case may be the vehicle which allows OOD to achieve universal applicability, particularly if it masks OOD complexities from the end user ...

*To Move Beyond a Metaphor, Reusability Needs to Get Real, Jerry Cashin, **Software Magazine**, 10/91*

Databases

... According to Chuck Williams, Inference cofounder ... "There have been several approaches to the integration of AI and databases that have been taken, and I see them continuing to evolve separately into the future. One approach is in embedding AI technology directly into the database. There are two major classes of this. One is in object-orientation (OO). The second is in providing more intelligence to query processing in database management systems, such that you can ask the database a set of questions that is larger than the specific data items that are stored"...

... Izzy Sobkowski, a leading AI consultant now on assignment at Con Edison in New York City, sees the mantle of object orientation becoming the framework for future systems. OO will not be just a programming technique under the auspices of knowledge-based systems. Rather, it will be the glue that binds the disconnected technologies. "The biggest problem in connecting knowledge bases to databases is that they don't talk clearly," Sobkowski said. The theory is that by subsuming both knowledge-based systems and database systems under an integrative OO framework, they will indeed talk the same language....

... [Dr. Kamran] Parsaye [president of IntelligenceWare Inc.] views knowledge-based systems as primarily high-level programming environments for information and knowledge-intensive tasks. Within this perspective a relational database can be closely integrated with a knowledge-intensive tasks. Within this perspective a relational database can be closely integrated with a knowledge base using relational predicates and objects. Parsaye contends that the relational predicate model and the object model can be fully interchangeable, forming the basis of a fully integrated intelligent database. Although Parsaye predicts that it will take five years for the concept of intelligent databases to infiltrate the mainstream, IntelligenceWare plans to announce the first intelligent database product within the next year ...

... According to [Tom] Hebert [of Computer Associates], "Knowledge-based systems is the merging of AI technology with database technology. The way it has to be done is to add OO capabilities to the existing database engine and to be able to integrate the repository information with the knowledge"...

*Surrounding Legacy Data with Intelligent Tools, Jessica Keyes, **Software Magazine**, 9/91*

to the next and more expensive stage. A successful completion may not, however, yield a single conclusion. At best, there are generally two alternatives that emerge and go on to the next stage: change to objects or continue as you currently are.

Stage 4. Pilot trial

The next step in the adoption process is to produce a live application using software engineers from the operational applications development group. The purpose is to apply object technology under normal operational conditions by developing and deploying a noncritical production application. Many companies wait until this stage to acquire and evaluate support tools for analysis, design, code management, repository management, deployment, and maintenance.

It is important that this project be done using the internal engineers whose normal job is developing the systems of the type to be implemented. A necessary prelude to this task will be the creation of a core of object experts within this applications development group. The pilot trial project is also used to test scalability. Therefore, this project should be several times larger than any previous project.

The pilot trial project will be a showcase for objects; therefore, careful selection is advised. One key purpose of this project is to achieve a success that is visible and relevant to the overall software development community within the company. This success must build sufficient confidence in object technology to motivate others to begin adopting objects themselves.

DEPLOYMENT PHASE

The deployment phase introduces objects into the mainstream of the company business. Objects should no longer be considered under test but, rather, ready for full production application. Before deployment can begin, all levels of management must be confident of the success and long-term viability of objects. A failure at this stage could adversely affect a manager's career or even the company's financial health.

Stage 5. Initial production

In the initial production stage, a strategically important application is implemented using objects and deployed for normal operation. This is no longer a trial; however, typically, there are still some open questions: Will objects provide the robustness, the resiliency, and the reliability needed to sustain an application over its life cycle? Will these qualities be enhanced or degraded after repeated upgrades and modifications? These questions can only be answered by continuous observation over a period of time and under the stress of normal operations.

Most often, the initial production project is undertaken by the same development team that implemented the pilot trial project. This application should have been identified as early as the beginning of the technology transfer phase. The requirements for this application should be considered part of the decision-making criteria during both stages 3 and 4.

Stage 6. Diffusion

Up to this point, the activities described here are carried on by the 20% of the software developers that are characterized as early adopters. The remaining 80% will not consider using any new technology until they have observed it used successfully by others. By emulating these successful applications, the late adopters can quickly achieve their own success for considerably less cost. This process of building success upon success is the typical manner by which most computer technologies become mainstream.

> ***The pilot trial project will be a showcase for objects ... One key purpose of this project is to achieve a success that is visible and relevant to the overall software development community within the company.***

CROSS-STAGE ISSUES

Each stage has different decision makers, different decision influencers, and different criteria for success. In the early stages, the decisions are made at a low level by the technical staff because initially the criteria and issues are technical in nature. In the middle stages, business issues must also be addressed. Therefore, decisions are made by a combination of middle and upper level technical management and business management. In the final stages, all of the technical issues should have been resolved; therefore, the decisions are based purely on business issues and criteria

The viability and maturity of the external commercial infrastructure that supports objects can become an issue in the adoption process. As a company progresses through the six stages of the adoption process, it has differing needs from outside suppliers. These needs begin in the prospecting stage with basic education and simple tools. In the middle and later stages, more and more sophisticated tools, services, and components are needed. One disadvantage of being an early adopter is that you may well reach a stage where the products and services that you need are not yet available because the market

has not yet developed sufficiently to support a commercial infrastructure. If this occurs, you have three choices: wait until the market develops, and the needed products and services are available; fill your own needs from an inside effort; or forge ahead by assisting the fledgling infrastructure companies in creating the products you need. The first approach is the least costly, and the second is the least risky in the short term. However, the third option will provide lower risk in the long term.

> ***It will require a significant amount of time and effort to gain sufficient understanding and experience with objects to trust them for critical applications.***

SECRETS FOR SUCCESSFUL ADOPTION

The most successful gatekeeper labs are ones that are problem driven not technology driven. The gatekeeper staff must acquire and maintain a good understanding of the problems faced by its clients in the operational groups. An effective gatekeeper lab needs to maintain constant interaction with the operational groups. This interaction should include listening to their problems and educating them on possible new candidate technologies. They must seek new technologies that specifically address the concerns of these clients. They must avoid selecting candidate technologies solely because the technical staff find them interesting. Each side of the technology transfer process must recognize and be sensitive to the motivation and constraints of the other side.

It is important at each stage in the process to identify specific goals to be achieved in that step. These goals should be formulated to address the key decision criteria for the next stage. If, upon completing a stage, there are still open questions and issues that prevent a clear decision at the next stage, then the endeavors of the stage just completed have fallen short of success.

Before moving beyond the prospecting stage, the gatekeeper lab should have one or more operational groups prepared to sponsor future activities. This sponsorship should include the loan of at least one engineer who will be a full participant in the proof-of-concept project. As the adoption process moves out of the exploration phase and through the technology transfer phase, this engineer on loan and one or more of the gatekeeper staff should remain with the program until the pilot trial is completed.

Do not let the press dictate how you run your business. The press can be a great source of information, but their business is to sell advertising and publications. A technology like objects can be overexposed by the press in the early stages of its development because it's new and exciting and will help sell publications. Sometimes this overexposure borders on hype, creating unrealistic expectations and uncertainty. Later, as the technology matures, the press may no longer consider it newsworthy and, therefore, it may suffer from underexposure. It is important to form your own judgements independent of what you see or don't see in the press.

SUMMARY

Even though object-oriented products and tools have been available from several sources for many years, very few major corporations have yet to embrace their use for production applications. For this reason, some software engineering managers are beginning to doubt if objects will ever become mainstream. Even the most avid supporters of objects will admit the adoption process has been slower then expected. There appear to be two reasons for this unhurried pace.

First, the expectation that objects would be quickly adopted was unrealistic. Objects are a complicated technology in which the underlying principles differ from those embodied in legacy software engineering methodologies. This change over in principles is the basis for the often referred to paradigm shift. For most, it will require a significant amount of time and effort to gain sufficient understanding and experience with objects to trust them for critical applications.

Second, in many cases, a company's lack of a well-defined technology adoption process has elongated this task. Technology adoption is far too important to the success of a company to be left as an ad hoc activity. It should be managed. To do so properly requires an effective methodology. Objects are a very valuable software technology that will, in time, become pervasive throughout the software industry. Those firms with an effective technology adoption process will be among the first to successfully exploit objects and they will be rewarded by improvements in their competitive position.

More to come ...

A sequel to this article will look at the pitfalls of the technology adoption process. It will draw examples from real life situations in anonymous companies to illustrate where the process can fail. It will discuss ways to avoid those pitfalls that can be avoided and how to deal with those that can't. ≡≡

substantially more difficult to fix," he says. Opposing object-oriented enhancements to Ada far less than Zeigler is Erhard Ploedereder, chairman of the Ada 9X Distinguished Reviewers and chief scientist at Tartan Inc. ... He says he isn't surprised at the criticism leveled on object-oriented programming by Zeigler and others. "There are two camps out there," he says. "One that says object-oriented design is snake oil, and the other that says it is essential and Ada ought to have it. There is no way to bring those camps together, so we need to tell the first camp if you don't want to use it, don't, and you will not have a performance penalty. To the second camp, we say we are supporting object-oriented design, and in a safer fashion than C++."

Why Ada Vendors Worry About Costs of Ada 9X, John Keller, ***Military and Aerospace Electronics,*** *9/91*

... Actor is an object-oriented language much like Smalltalk, and the syntax resembles that of C in some cases and Pascal in others.... There are some disadvantages to using Actor, however. Actor is an language that prides itself on being "theoretically correct;" it forces you into the object-oriented mold even when you may not want to be there. Actor's advertising claims that you can develop applications faster in Actor than in other languages, and that may be true once you are accustomed to writing object-oriented code; but the learning curve to get there is steep. Really steep. As a consequence, Actor as a language has not taken the industry by storm. Another disadvantage of Actor is that is uses a lot of memory and in certain cases it is slow. As I understand it, Whitewater is having more success with its add-in products, especially its excellent Resource Toolkit (which I use)....

How to Write Your Windows Applications, Chris Cleveland, ***The Chicago Computer Guide,*** *9/91*

... The most talked about new feature [in Version 2.0] is Borland C++'s support for developing Windows programs. Unlike Microsoft's and Zortech's C and C++ offerings, Borland's C++ V2.0 contains everything necessary to build custom Windows applications in short order ... After installation, you can build your first Windows application from the provided examples with only a few keystrokes from inside the integrated environment ... Borland C++ Version 2.0 is an excellent performer and an exceptional value. Whether you're building simple command line utilities or multimedia applications, you can be confident that choosing Borland C++ toady won't be an mistake tomorrow.

Product Review: Borland C++ Version 2.0, Scott Moody, ***Object Magazine,*** *9-10/91*

... C++ is truly a better C. For one, it implements stronger type checking through the specification of argument types in function prototypes...Very few developers were willing to have a large project based on a language supported by only a single company. The emergence of C++ as the de facto object language for UNIX and DOS systems removed this factor. Almost all of the new generation of ODBMS are based on C++. They use standard compilers, standard debugging tools, and standard libraries. C++ programming environments, like Saber-C++, now also support the use of ODBMSs...C++ is clearly emerging as the object programming language of choice ... but new tools for C++ are essential to make the transition. The C++ software engineer needs language-specific tools for all development tasks ...

Moving from C to C++, David R. Reed, ***Object Magazine,*** *9-10/91*

... Many organizations are concerned with how they leverage old skills into the new object environment. Some help may come in the Object SQL query language. Object SQL can be learned easily by someone who knows SQL. Like SQL, Object SQL is a nonprocedural language for defining and manipulating databases. Unlike SQL, it can access object databases. Some implementations will also enable Object SQL to access relational databases and will be used in conjunction with SQL Gateway products. One of the most attractive characteristics of SQL has been that it is fairly standard across many vendors' relational DBMSs. By contrast, there is as yet no standard for Object SQL ...

Objects and SQL: Accessing Relational Databases, Mary E. S. Loomis, ***Object Magazine,*** *9-10/91*

Antihype

... [Bill Gates]: It's important for me what kind of programs it will enable. Certainly over half the new things going on at Microsoft relate strongly to object orientation. But people act like it's fairy dust without talking through the scenarios ...

Gates' Vision: Putting Archives to Use, John Schneidawind, ***USA Today,*** *10/9/91*

Paradigm Shift

... Building a single, integrated model for the problem domain is something the securities industry has to do. We're face to face with the complexity of the solution right now. Other industries won't be far behind. Take a close look at your own problem domain; you may find that the celebrated paradigm shift is not a problem of changing the way people think but of dealing with the resulting solution ...

The Complexity of the Solution, Bill Welch, ***Object Magazine,*** *9-10/91*

... There is an attitudinal problem. There is much talk about the software manufacturing paradigm changing. The real paradigm shift needed for the 90s is to change the actual definition of computing. Industry thinking is still handicapped by its descent from the punched card. The sequential, procedural approach — even when timesliced or executed in massively parallel chunks of work — has served its time ...

Distributed Object Management, Herb Osher, ***Object Magazine,*** *9-10/91*

continued on page 14 ...

FYI

Excerpts from leading industry publications on aspects of object technology

Graphics

... But in a world increasingly jammed with OOP proselytes, we still don't have an OOP graphics front end for these [graphics] libraries. I would like to see something that would give me ONE Object Oriented Design perspective with support for several graphics libraries ...

Graphic Developer's Taste Test, William E. Gates, ***Midnight Engineering,*** *10/91*

Industry News

... [IBM's James Cannavino] did concede that Apple had more code to show for object-oriented programming, though he indicated that his partner had not incorporated the technology into a corporate-wide environment. "Apple's written more code," Cannavino said. "I think we have a better enterprise design"...

Apple, IBM Ink Formal Alliance Plan, John Thompson and Margaret Ryan, ***Electronic Engineering Times,*** *10/7/91*

... Objects were the subject of much debate, both public and private, at Agenda [92]. Many of the speakers — from Sun, Microsoft, NeXT, Light Source Inc. and Hewlett-Packard — traded barbs about how their objects were better than the other fella's and charging that they're all being penalized by vendors and the media throwing around the term "object-oriented" as if everyone understood what it meant ...

The Fabulous Feline Dishes the Choicest Dirt from Agenda 92, BRB, ***PC Week,*** *9/30/91*

Operating Systems

... The more advanced pen-computing operating systems use object-oriented design for memory management. In contrast to desktop GUI applications, which may require multiple megabytes of memory, object-oriented applications, typically require only about 100K to 200K because the operating system conserves memory by eliminating redundant code ...

Is the Pen Mightier?, Kathleen Melymuka, ***12A-550 CIO,*** *9/15/91*

Language & Tools

... To characterize Turbo Vision as only an interface builder is to shortchange it. Much of the product's benefit occurs behind the scenes, where Turbo Vision truly delivers a strong skeleton for complicated applications. Turbo Vision is not a trivial product to learn, as it combines object orientation, event-driven programming, and complicated interface logic with multiple resizeable windows, hotkeys, resources, and mouse support. Successfully uncoupling the interface from program logic requires a complete switch to the event-driven paradigm — a switch that may be as difficult as the move to object orientation. And, as Borland says in the documentation, "If you use any of [Turbo Vision] at all, you should use all of it"... For those who are making the big move from traditional character-based programs to a GUI future, Turbo Vision is perfectly placed. You don't want to tackle event-driven programming, application frameworks, and an unfamiliar GUI API simultaneously! Even if you aren't planning on moving from the character-based world of DOS, you probably are developing more and more complicated programs and your users are becoming more and more demanding about ease-of-use issues. Turbo Vision, once learned, gives you a tremendous edge ...

... Although I embrace the results of the switch to objects and events, I can't help but think that a major reason for C++'s success is that it allows gradual adoption of object orientation and that an Application Framework allowing gradual adoption of event-driven programming would be similarly embraced by programmers ...

... If you have an object that actually contains other objects, you generally ... Well, you generally wish there was an object-oriented database available for less than several thousand dollars. Now there is. Although the exact pricing and marketing details were still unclear, the German company BKS Software GmbH plans to open an office in the US and begin distributing its object oriented database, POET, for MS-DOS, Windows, and the NeXT machine by the time you read this ... The product, as I saw it in beta test, had some minor problems with documentation and the makefiles for the demonstration programs. Otherwise, it worked flawlessly, allowing me to store and retrieve complicated objects with little effort. I only used relatively small, non-network objectbases, so I can't vouch for the performance characteristics in that area where object databases claim the most — very large data stores (pictorial data, for instance), Still, when it hits the street in the US, it will be a real bargain.

Inherit the Window, Larry O'Brien, ***Computer Language,*** *9/91*

... Despite assurances that 9X enhancements will be affordable, some Ada experts still aren't convinced. Stephen Zeigler, vice president for development ... at Verdix Corp., says the proposed object-oriented-design changes represent the most substantial costs to 9X implementation, which he says will be "in the millions of dollars, including documentation" ... Zeigler claims the object-oriented features, despite their popularity among software engineers, are what ultimately will stick it to compiler vendors and their customers. "Object-oriented design tends to hide problems until implementation time, and changes are

MULTIMEDIA

Incorporating graphical content into multimedia presentations

by Louis Berman and Kenneth Weiss

Multimedia applications use a diverse range of media for data input and output.

Whereas a traditional application might interact with a user only through a keyboard and a simple display, a multimedia application might add a touch screen, high-resolution graphics, full-motion video, and full-stereo sound. This article will begin to discuss the various types of devices used in a multimedia production.

It should, however, be realized from the onset that there is no one "multimedia machine." In most cases, a standard computer with additional software and/or peripherals added in will do. That is not to say that any machine can be made into an effective multimedia platform. Multimedia implies a broad communications band width that, in turn, requires a substantial amount of computing power. In general, these types of applications are very memory hungry and disk space intensive.

CHARACTER-BASED VERSUS BITMAPPED GRAPHICS

An obvious and useful multimedia peripheral is a high-quality color display. In almost all cases, this means a bitmapped graphical display; but, to assert this "obvious" choice with any level of confidence, one needs to look at the alternative.

Character-based displays have been around for a number of years. These displays tend to feature a finite set of characters (i.e., letters, numbers, and symbols) that are arrayed within a fixed grid arrangement using a limited set of colors, a few display attributes, and a single font. In the industry standard VGA text format, this means 25 lines of 80 characters each, where each element can display only 1 of 256 possible characters at a given time using only 1 of 8 background and 1 of 16 foreground colors.

At best, this offers a limited range of visual possibilities. I'll grant you that a number of creative programmers have stretched the medium to its limits — we've all seen crudely designed "graphics" on character-based displays — but the results are usually less than satisfactory.

On the other hand, bitmapped graphical displays can be used to draw very realistic pictures. This is accomplished by changing the smallest display unit from an entire character to a single pixel (or dot). These tiny dots form pictures in the same way as a pointillist painting or a halftone picture in a newspaper. Viewed up close, it's easy to resolve individual pixels; but, as one pulls further back, the dots merge to form distinct and recognizable picture elements such as lines, circles, shadows, textures, and the like.

A typical bitmapped PC screen has 640 pixels across and 480 pixels down for a total of 307,200 pixels. By selectively turning individual pixels on or off, it is possible to draw sophisticated colorful graphics. With this scheme, we can even display text in several different fonts and/or sizes along with a diverse range of graphical elements placed anywhere on the page. A good example of the dramatic difference bitmapping makes is to compare a character-based DOS application, such as Lotus 1-2-3 with a Windows-based bitmapped graphical application like Microsoft Excel. At a glance, Excel looks slick and capable, whereas 1-2-3 looks boxy, outdated, and thoroughly uninspiring.

A second and equally important aspect of bitmapped graphics is their ability to display each pixel in a separate color. By alternating the color of individual pixels, it is possible to produce a wide spectrum of colors and effects. Indeed, true onscreen photorealism is now within the grasp of the modestly well heeled.

All this good stuff, however, comes at a price. It takes a lot more system resources (i.e., bigger hard disks, expensive displays and display adapters, etc.) to store and display a bitmapped picture than it does to display a character-based one. Using the character-based display described above, a full screen could be stored in only 4,000 bytes — 2,000 bytes for the data and 2,000 bytes for the color and display attribute information. On the other hand, a 640 × 480 pixel 16-color bitmapped picture would

Louis Berman and Kenneth Weiss are principals of Braintree Ltd., a Bayside, NY based consulting firm specializing in multimedia and database applications.

require 153,600 bytes. And 16 colors is the absolute minimum that should be used for multimedia work. Actually, we consider 256 colors to be barely acceptable. Of course, that doubles the storage requirements to 307,200 bytes for each screenful of information.

And, to provide true photorealism, a lot more colors are required — 16.7 million more of them, to be precise. This is achieved by using 24 bits per pixel which, unfortunately, triples the picture storage size to 921,600 bytes. That's a lot of data, but the pictures look great. Then again, a 640 × 480 × 24 bit display is rather modestly sized for today's world. Just imagine a 1,280 × 1,024 × 32 bit display (with 8 bits used for color transparency). That adds up to a whopping 5,242,880 bytes per screenful. Wow! Just imagine how hard your hard disk would have to chug to pass that data to the display adapter in a timely fashion. Needless to say, such displays severely tax the abilities of current desktop technology. The result, though, is absolutely superb. And data compression can be used to greatly reduce this onerous requirement — quite often by a factor of 30 to 1 or more.

In summary, while it is possible to produce a visually limited multimedia application using a character-based display, it is almost always better to do it on a bitmapped display. Moreover, it is better to use a higher resolution display, budget and storage requirements permitting.

VIDEO INPUT

Most computers display information by sending data that is stored on a hard disk directly to a video display adaptor. The adaptor's job is to take the computer information and translate it into a signal that the monitor can use.

One very powerful technique used in multimedia presentations is to display not only computer-generated data but to display input from video devices such as television receivers, VCRs, video cameras, and/or laser disks. These devices are NTSC compatible. NTSC is the electronic standard used by our televisions. Displaying NTSC signals on a computer display requires additional equipment to merge it with the data coming from the computer since standard video adaptors cannot accept NTSC signals as input. A special video overlay board that has connectors to physically plug in the NTSC devices is required. A number of video overlay boards also support a wide range of image manipulations including color correction, line smoothing, contrast control, and the like — all of which can be performed under direct computer control. By combining the NTSC video with the computer-generated video, it's easy to add special effects such as titles or chromakeyed overlays to presentations. Just imagine being able to merge any video footage with any computer-based graphic and/or animation. You could, for instance, create your own onscreen weatherman.

These boards used to cost $2,500 and up. Recently, though, the boards have broken the $700 dollar range, and we expect the price to continue to drop as time goes on. Similarly, the software that controls the boards has dropped into the low hundreds. I might also add at this point that the entire process has been greatly simplified. You used to need a Ph.D. to operate one of these things. Now, you simply need an extraordinary amount of patience.

Thankfully, though, the effort is generally well rewarded. For starters, NTSC devices are both cost effective and all pervasive devices. Video is everywhere — so much so that people are starting to expect it to appear on their computers. Actually, many corporations already have vast libraries of live video that they wish to present in an interactive format using computers. All it takes is a video overlay card and a computer-controlled VCR or videodisc player.

STATIC IMAGES

One of the challenges of any multimedia presentation is gathering the content. A very effective way to produce content is to use a video overlay board to capture video images as computer files. Most video overlay boards have facilities to grab a video image and write it onto a standard disk in a number of graphics formats. Once captured, those images can be incorporated into any presentation.

It is relatively simple to plug the NTSC device into the overlay board and capture a picture. Images can be taken directly from a television or camcorder. These are very common, inexpensive devices that are readily available. One of the newest NTSC devices is the digital still frame camera. It is a cross between a regular photographic camera and a video recorder. It records the still photo on a miniature floppy disk in the camera. The pictures are then viewed by playing them back on a television. Since digital cameras are also NTSC compatible, it is a very convenient way to collect information for multimedia presentations.

FULL-MOTION VIDEO

Most people think of multimedia presentations as full-motion video. While professionally shot full-motion video is the ultimate in multimedia, it is often very costly (more than $2,000/minute) and complicated to produce. Most multimedia productions do not have or even need full-motion video. But, if it is necessary, the laser disk is the device of choice (note: CD ROM players are poised to replace laserdiscs in this role).

Laserdiscs have been available for many years. They are relatively inexpensive at under $1,000. The media (laserdisc) looks like a large audio CD and has 30 minutes of video stored on each side. The laserdisc can output full-motion video and high-quality stereo sound into the video overlay board. The most useful feature of the laserdisc is that the computer can instruct the laserdisc where to start playing and when to stop playing. This feature (called frame addressing) allows sophisticated playing sequences to be constructed. For instance, an instructional segment can be displayed in a computer-based training piece. The computer can then ask the student to answer a question. If the student gets the answer wrong, it is very easy to replay the previous sequence or, better yet, to play an additional instructional video segment to better explain and reinforce the material. This can only be done because the videodisc can be addressed to play from a specific starting place (frame) on the videodisc. A downside to videodisc is that it is expensive to get the video content onto the disk. At present, it costs from about $1,100 to $1,800 to master a disc. Obviously, this, along with the other costs, tends to place videodisc use within the hands of the reasonably well off. One thing to remember: once the videodisc is mastered, it cannot be changed without remastering the disc (for another $1,100 to $1,800)!

VCRs can also be hooked into a video overlay board. The quality of picture is generally less than that of a laserdisc and tends to deteriorate over time. Another major difference is that most VCRs do not have frame addressing. This makes it difficult to produce sophisticated play sequences. The basic VCR functions may be available (start, stop, rewind, etc.); however, precisely playing sequences of video is often not possible. Also, VCRs are sequential devices. To play a video sequence at the end of the tape dictates that the tape be fast forwarded to the end. In contrast, a laserdisc can be directly positioned without a long wait period. This is analogous to data access from magnetic tape vs. hard disk. The major advantage to VCR-based multimedia productions is that most people have VCRs, making playback access easy and inexpensive.

NON-NTSC IMAGES

While input from NTSC devices is very useful, the vast majority of multimedia content comes in from other sources. These sources include clip art libraries, drawing programs such as CorelDraw and PC Paintbrush, and authoring systems like ToolBook, scanners, and CD ROM players.

Graphics are transferred and stored in a variety of formats. The most popular being TIFF, GIF, WMF, PCX, and BMP. Most graphics programs can convert their own image formats to one or more of the above. There are also a number of standalone conversion programs that can take the graphic data files and convert them into another format. The standardization of graphic file formats makes it relatively easy to exchange graphics. Indeed, the video capture boards usually make it easy to capture an image into a variety of graphics file formats. The following modalities are all able to import and/or export in standard graphics formats. It is important to remember that the following devices do not need a special overlay board to work.

Scanners

Scanners come in various flavors. The major types are flatbed, full-page and hand-held scanners. Hand-held scanners transfer any image into a standard graphics file format by physically moving the scanner over the page. Most hand-held scanners are four inches wide. They are quite useful, especially for smaller images such as logos and signatures. If a larger picture must be scanned, it is sometimes possible to scan the picture in multiple pieces and join them. The process can be tricky, although it is possible.

For full-page scanning, a flatbed scanner is best. The image is placed on a glass plate (much like in a copy machine) and transferred into a graphics format and placed in a disk file. The scanned images frequently need some form of color correction and/or editing. The scanned images can be brought into various paint programs to correct a variety of problems and scanning artifacts. Also, take note of the fact that scanned text is treated as a picture. You cannot read this information directly into a word processor. There are, though, optical character recognition programs that can convert scanned text into machine-readable characters. Page scanners scan a page at a time in a manner similar to the way a fax machine scans a page. Because page scanners scan one page at a time, it is impossible to use them to directly scan pages that are bound within a book.

Paint Programs

Sometimes it is necessary to create computer-based art from scratch. These programs give the artist the necessary tools to draw an amazing variety pictures. It is often necessary to read clip art into the paint program to modify or edit the original picture. As with scanners, paint programs import/export in a host of graphics formats.

Authoring Systems

Besides collecting images, it is necessary to weave the individual pieces into coherent presentations. There are many multimedia authoring systems such as Toolbook and Authorware. Each package provides a set of drawing tools to create graphics as well as tools to manipulate graphics created by other packages. It is not uncommon to scan an image, import it into an authoring system, and clean or alter it with the authoring system's own tools.

CONCLUSION

Although we have discussed only some of the ways that graphical content may be incorporated and displayed in multimedia presentations, we hope you have gained insight into how graphics can enhance your presentations. And, as the technology matures, new approaches will become available. ≡≡

≡ Your Turn ≡

How about it? How are we doing? This is the place to send us your message. Tell us what you like and what you don't, what you'd like to see but don't — or whatever other feedback you'd like to send. Send comments to Marie A. Lenzi, 155 E. 31st St. #22, NY, NY 10016; fax to THE HOTLINE, *(212)725-0224; or, if you're ordering, send along with your order.*

Please check whichever box applies:

	Read entirely	Scanned	Found helpful	Will refer to	Didn't read
Feature	❑	❑	❑	❑	❑
Getting Started	❑	❑	❑	❑	❑
Objects down under	❑	❑	❑	❑	❑
FYI	❑	❑	❑	❑	❑
Product Announcements	❑	❑	❑	❑	❑
Calendar	❑	❑	❑	❑	❑

Please rate the following **1** (dislike or strongly disagree) to **10** (like or strongly agree):

Presentation of material __ Comments: _____

Accuracy of material __ Comments: _____

Overall helpfulness in your job __ Comments: _____

What else would you like to see in THE HOTLINE? _____

Any other comments? _____

_____

_____

Name, title & phone (optional): _____

hotline on
OBJECT-ORIENTED
technology

❑ **Yes,** plug me into the latest thinking and developments in object-oriented technology. Enter me as a subscriber at the term marked below and rush me the current issue. This is a risk-free offer — I may cancel my subscription at any time and promptly receive a refund for the unused portion.

1 year (12 issues) ❑ $249

2 years (24 issues) ❑ $478 *(save $20)*

(outside US add $30 per year for air service)

Back issues @ $25 each *($27.50 foreign)*:
Vol.1, Nos. _____
Vol.2, Nos. _____ Vol.3, Nos. _____

❑ **Phone order**
Call Subscriber Services
(212)274-0640

❑ **Fax order**
Fill out term and name/address information, then fax to:
(212)274-0646

❑ **Send me a copy of**
THE INTERNATIONAL OOP DIRECTORY
@ $63 ($71 foreign)
(all directory orders must be prepaid; fax or mail credit card information to SIGS, 588 Broadway, #604, NYC 10012; make check payable to OOP Directory. Foreign orders must be prepaid in US dollars drawn on US bank.)

D1NC

❑ **Bill me**
❑ **Check enclosed**
❑ **Credit card orders**
❑ MasterCard
❑ Visa

Card# Expiration Date

Signature

Name

Company

Street/Building#

City/Province

ST/Zip/Country

Telephone

D1NC

Mail (and make out check, if applicable) to:
THE HOTLINE
Subscriber Services
P.O. Box 3000, Dept. HOT
Denville, NJ 07834
(foreign orders must be prepaid in US dollars drawn on a US bank)

Send me a complimentary copy of your related publication(s):

❑ THE JOURNAL OF OBJECT-ORIENTED PROGRAMMING

❑ OBJECT MAGAZINE

❑ THE C++ REPORT

Call Subscriber Services
(212)274-0640
or send order to
SIGS, 588 Broadway, #604
New York, NY 10012

D1NC

hotline on
OBJECT-ORIENTED
technology

VOLUME 3, NUMBER 3 | THE MANAGER'S SOURCE FOR TRENDS, ISSUES & STRATEGIES | JANUARY 1992

Enterprise object modeling: knowing what we know

When philosophers first asked "How do we know what we know?" their query reflected awe at participating in the definition of the human condition.



Robert Shelton
as kibitzed by Marie Lenzi

Our predecessors sought to give meaning to being human, to understand thinking and knowledge, because they believed that brain power distinguished humans from lesser animals. With our brains, we could create, we could use tools, we could compute…The future was ours!

The plight of modern man reminds me of the line from Dr. Seuss, "Look Ma, no hands!" Today, we find ourselves drowning in the products of our brain power: computer technology—everywhere! This one creation has forever redefined the human condition. As a writer, all my work is done on a computer that my colleagues will tell you is permanently attached to the ends of my fingers! I can't even wash my clothes or drive my Nissan pocket-rocket without using a computer. We could reasonably surmise that the human condition is no better off for the chip. We're still asking "How do we know what we know?" However, our focus has shifted away from ourselves.

Today's computerized enterprises are drowning in data, a fate made possible by the chip. Not only do we not know what we know, we usually don't know what it means when we do know it. So separated are our megabytes and programs from the businesses they were intended to support that what we think we know often means something else! We are not living in the information age. Information implies meaning. We are living in the data age. Data implies quantity, not quality. Today's organizations struggle with the tools and techniques of semantic modeling not for the joy of enlightenment, but in a desperate grab for meaning. What is sought is the tool for imparting meaning to our torrents of data.

Good tools exist for capturing the meaning of this flood of data, but few information professionals really use them. Enlightened practitioners (Fleming and Von Halle in their *Handbook*, Bob Brown and Tom Bruce of IDEF1X notoriety, and Dan Tasker, with *4th Generation Data*) have given us concise methods for extracting meaning and data structure at the enterprise-wide level. But all too many information systems managers and practitioners think the process of enterprise modeling gets in the way of meeting programming deadlines. Buildings aren't

continued on page 3…

IN THIS ISSUE

FEATURE — Enterprise object modeling: knowing what we know *Robert Shelton*	1	2	FROM THE EDITOR
GETTING STARTED — Adopting objects: pitfalls *J. Michael Connell*	5	16	PRODUCT ANNOUNCEMENTS
OBJECTS DOWN UNDER— Adoption rate of object technology: a survey of NSW industry *B. Henderson-Sellers*	8	18	NEW PARTNERINGS & ACQUISITIONS
FYI — Industry publication excerpts	14	19	CALENDAR

From the Editor



"The time has come, the Walrus said..."

This is my last *Hotline* editorial. The time has come for me to say so long. The time has come for me to pass editorial responsibilities for the *Hotline* on to another. The time has come for a newsletter that truly targets the strategic level of the industry.

When I took over as editor of the *Hotline* in July, 1990, objects were still very much a niche technology. There was no management-level periodical targeted specifically at objects. The *Hotline* filled that gap, but at an operational level: at the level of development and project managers, those who truly needed immediate assistance. Now, in less than two years, object technology has expanded to a point where we can take the *Hotline* to the level for which it was originally intended. However, it has to go there without me.

While the industry was getting ready for a strategic newsletter in objects, I was building my own object technology business, as well as launching *Object Magazine*. Now, much to my dismay, I find myself in a position where I have to make a choice: either increase the hours in a day or give up a responsibility. Why can't I get control over the laws of physics?

We searched for quite a while to find the correct person and, of course, he was right under our noses. **Robert Shelton** has been a friend and colleague for many years. You get a firsthand look at his style and approach this month, as he is our feature author. However, you won't get the complete picture. In addition to his interest and skill in strategic level issues, he is an accomplished technician. (He regularly contributes product and book reviews to *Object Magazine*.)

Mr. Shelton feels as strongly as I do about closing the "Business/Technology Gap." His initial article, "Enterprise object modeling: knowing what we know," reflects a need for serious work among business and technology types. Pooling intellectual resources to re-engineer business (not just business applications) and to incorporate technology as an integral part of the restructured organizations requires a mix of strong business and technical skills, acute written and spoken communication skills, and, mostly, a reigning in of individual ego (and all I want for Christmas...). Robert and I will be doing work in this area, as evidenced by the co-byline on his article. We plan joint writings and conference presentations on the topic, the first to be presented at Object Expo, June 1–4 in New York.

There are other things going on in this month's *Hotline*. In the December '91 issue, **Mike Connell** offered "Adopting Objects: a Path." This month, part 2 of the series offers "The Pitfalls." Mike recognizes the fact that objects is a "team sport," that there are nontechnical as well as technical considerations to be addressed, and that without a direction or a target objective, we can get as bogged down in R&D as did the dinosaurs in Le Brea.

And, finally, this month **Brian Henderson-Sellers** and **J. M. Edwards** have looked at the adoption rate of object technology in Australian industry. I am pleased that Brian was able to provide an article for my last *Hotline*. While I have been editor, he has done a fine job of keeping us informed on what and how objects are doing in Australia.

Well, the time has come to say so long from here, but not goodbye. The time has come to wish the best to Robert and to the continuing growth and maturation of the *Hotline*. And the time has come for the industry to finally have a strategic level newsletter in object technology.

"The time has come, the Walrus said, to speak of many things..."

Editor
Marie A. Lenzi

SIGS Publications, Inc. Advisory Board
Tom Atwood, *Object Design*
Grady Booch, *Rational*
George Bosworth, *Digitalk*
Brad Cox, *Information Age Consulting*
Chuck Duff, *The Whitewater Group*
Adele Goldberg, *ParcPlace Systems*
R. Jordan Kreindler, *General Electric*
Meilir Page-Jones, *Wayland Systems*
Tom Love, *Consultant*
Bertrand Meyer, *Interactive Software Engineering*
Sesha Pratap, *CenterLine Software*
P. Michael Seashols, *Versant Object Technology*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

Hotline Editorial Board
Jim Anderson, *Digitalk, Inc.*
Larry Constantine, *Consultant*
Mary E.S. Loomis, *Versant Object Technology*
Reed Phillips, *Knowledge Systems Corp.*
Trygve Reenskaug, *Taskon A/S*
Zack Urlocker, *Borland International*
Steven Weiss, *Wayland Systems*

SIGS Publications, Inc.
Richard P. Friedman, Founder & Group Publisher

Art/Production
Kristina Joukhadar, Managing Editor
Susan Culligan, Creative Director
Kristin R. Juba, Production Editor
Caren Polner, Desktop Designer

Circulation
Diane Badway, Circulation Business Manager
Kathleen Canning, Fulfillment Manager
John Schreiber, Circulation Assistant

Administration
David Chatterpaul, Bookkeeper
Suzanne W. Dinnerstein, Conference Manager
Laura Taylor, Conference Coordinator
Jennifer Fischer, Assistant to the Publisher
Amy Stewart, Projects Manager
Jennifer Englander, Administrative Assistant
Margherita R. Monck, General Manager

Jane M. Grau, Contributing Editor

hotline on OBJECT-ORIENTED *technology*

The Hotline on Object-Oriented Technology (ISSN #1044-4319) is published monthly by SIGS Publications, Inc., 588 Broadway, NY, NY 10012, (212)274-0640.

© Copyright 1991 SIGS Publications, Inc. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the U.S. Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher. Mailed First Class. Subscription rate — one year (12 issues) $249, Foreign and Canada $279. Single copy $25.

POSTMASTER: Send address changes & subscription orders to The Hotline, Subscriber Services, P.O. Box 3000, Dept HOT, Denville, NJ 07834.

Submit editorial correspondence to Marie A. Lenzi, 155 East 31st Street, #22S, New York, NY 10016. voice: (212)213-8701, fax: (212)725-0224

SIGS PUBLICATIONS

Publishers of *Hotline on Object-Oriented Technology, Journal of Object-Oriented Programming, Object Magazine, The X Journal, The C++ Report, The Smalltalk Report,* and *The International OOP Directory.*

Back issues available

Vol.3, No. 1/November '91 ≡ Leading the U.S. semiconductor manufacturing industry toward an object-oriented technology standard • *Hollowell* ≡ Coping with complexity: OOPS and the economists' critique of central planning • *Lavoie, Baetjer, & Tulloh* ≡ Choosing Object Technology: What's the object? • Rosenthal ≡ OOP: the MISsing link • *Rovira*

Vol.2, No. 12/October '91 ≡ A modest survey of OOD approaches • *Bulman* ≡ What is a "certified" object programmer? • *Bellin* ≡ Perspective: investing in objects today • Bowles ≡ Object oriented in Melbourne, Australia • *Haebich* ≡ The Object Management Group • *Guttman & Matthews*

Vol.2, No. 11/September '91 ≡ From applications to frameworks • *Urlocker* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Getting started with object technology: efffectively planning for change • *Lorenz* ≡ Object statistics on the way • *Lenzi* ≡ On objects and bullets • *Onarr*

Vol.2, No. 10/August '91 ≡ Distributed object management: improving worker productivity • *Osher* ≡ Getting the best from objects: the experience of HP • *Coleman & Hayes* ≡ Applications: EC employs object technology ... • *Stein* ≡ Capacity Planning: Fiddling while ROMs burn • *Rovira*

Vol.2, No. 9/July '91 ≡ Multimedia is everywhere! • *Weiss & Berman* ≡ Developing an object technology prototype • *Kulinek* ≡ Object-oriented capacity planning • *Rovira* ≡ How OOP has changed our developmental lifecycle • *Zeik* ≡ Modularization of the computer system • *Angenius*

Vol.2, No. 8/June '91 ≡ Domain of objects: the Object Request Broker • *Dyson* ≡ Object-based approach to user documentation • *Reeder* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Do we need object-oriented design metrics? • *Hopkins*

Vol.2, No.7/May '91 ≡ Hybrid object-oriented/functional decomposition for software engineering • *Henderson-Sellers* ≡ So, what makes object databases different? (Part 4) • *Blakey* ≡ Using the generic application to solve similar domain problems • *Gossain* ≡ Experiences using CLOS • *Hopkins* ≡ International Conference on Object-Oriented Technology, Singapore • *Bennett*

Vol.2, No.6/Apr. '91 ≡ An artist's perspective of programming • *Rovira* ≡ So, what makes object databases different? (Part 3) • *Blakey* ≡ Moving from Pascal to C++, Part 3 • *Gole* ≡ Object projects: what can go wrong • *Stewart* ≡ Reflections from LOOK-'91 • *Anders-Angenius*

Vol.2, No.5/Mar. '91 ≡ Large-scale object systems development • *Morgan & Davis* ≡ Data definition languages • *Blakey* ≡ Object disorientation • *Kerr* ≡ Moving from Pascal to C++, Part 2 • *Keller* ≡ The Object Management Group • *Barber*

Vol.2, No.4/Feb. '91 ≡ Moving from Pascal to C++ • *Sollisch* ≡ The Australian object-oriented scene • *Henderson Sellers* ≡ So, what makes object databases different? • *Blakey* ≡ Managing the evolution • *Novobilski* ≡ 1990: the year of realism • *Love* ≡ Choosing a configuration for object development • *Moody*

Vol.2, No.3/Jan. '91 Special issue: CASE and Methodology ≡ Object orientation: stop, look and listen! • *Page-Jones* ≡ The spiral model for object software development • *Wasserman & Pircher* ≡ Bootstrapping object-oriented CASE • *Wybolt* ≡ A CASE tool for object software development • *Ackroyd* ≡ Comments on "Race for the database pennant" • *Loomis*

Vol.2, No.2/Dec. '90 ≡ Waiting for OODB2 • *Durham* ≡ Object modeling • *Singh* ≡ Object-oriented COBOL — the next generation • *Adams & Lenkov* ≡ Object-oriented training in the UK • *Hopkins*

Vol.2, No.1/Nov. '90 ≡ Objects by teamwork • *Constantine* ≡ Race for the database pennant: a third-generation manifesto • *Stein* ≡ Project team descriptions *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.12/Oct. '90 ≡ Crown jewels: the value of OOT in MIS departments • *Bonar* ≡ Why is an object database faster than a relational system? • *Loomis* ≡ Visual programming object language environments • *Klausner* ≡ Hype — necessarily (or a necessary) evil? • *Adams* ≡ On Managing: Déjà Vu • *Lenzi*

Vol.1, No.11/Sep. '90 ≡ Introducing OOPS through design — it's all in the words • *Copen Goldstein* ≡ The Australian object-oriented scene • *Henderson-Sellers* ≡ Objects on tour (Review of Borland's OOP World Tour) • *Lenzi* ≡ Design principles for modular product development • *Bennett* ≡ Asset management and object-oriented technology • *Hopkins & Warboys*

Vol.1, No.10/Aug. '90 ≡ WyCASH+: An application built within an OOP environment • *McCullough & Deshler* ≡ Mathematics and methodology • *Martin* ≡ Organizing for reuse • *Goldberg & Rubin* ≡ CALS Expo–89 Conference • *Purdy*

Vol.1, No.9/July '90 ≡ Cognos and Eiffel: a cautionary tale • *Leathers* ≡ What's in an object anyway? • *Novobilski* ≡ The impact of Windows 3.0 on the OOP market • *Urlocker* ≡ Introducing reuse into the development process • *Goldberg & Rubin*

Vol.1, No.8/June '90 ≡ A taste of the USENIX C++ Conference • *Wells* ≡ Comparing relational and object database management systems part 2 — Composites, flexibility, and extensibility • *Martin* ≡ Case studies in prototyping, part 3 • *Goldberg & Rubin* ≡ Object-oriented requirements analysis • *Berard*

Vol.1, No.7/May '90 ≡ Object-oriented development: not business as usual • *Lorenz* ≡ Comparing relational and object database management systems part 1 — Single paradigm programming is the key difference • *Martin* ≡ Case studies in prototyping, part 2 • *Goldberg & Rubin* ≡ Understanding the recursive/parallel lifecycle • *Berard*

All back issues of *Hotline on Object-Oriented Technology* are available. Please call (212)274-0640 for details.

To order, see back page; for reprints of individual articles, call Michael Biggerstaff at Reprint Management Services (717) 560-2001.

Calendar

The Hotline Calendar presents conferences and meetings that focus exclusively on object-oriented technology. To have a meeting or conference listed, please send the dates, conference name and location, sponsor(s) and contact name and telephone number to the Editor: Marie A. Lenzi, 133 East 31st Street, #22S, New York, NY 10016, fax (212) 725-0224.

February 3–5, 1992	February 5–7, 1992	February 11–21, 1992	February 25–28, 1992	March 23–26, 1992
LOOK '92	**OOP '92**	**C++ Across America**	**Software Development '92**	**DB/Expo '92**
Copenhagen, Denmark	Munich, Germany	SF, LA, Dallas, Chic., Bos., NYC, Wash. D.C.	Santa Clara, CA	San Francisco, CA
Sponsor: Niveau, *Object Magazine*, & *The Journal of Object-Oriented Programming*	Sponsor: *The Journal of Object-Oriented Programming & Object Magazine*	Presented by: *The C++ Report*	Sponsor: Miller Freeman	Sponsor: NDN
Contact: Miles Silverman (212) 274-0640	Contact: Miles Silverman (212) 274-0640	Sponsored by: Microsoft	Contact: (415)905-2414	Contact: (415)941-8440
		Contact: (212)274-9135		

New Partnerships and Aquisitions, January 1992

KnowledgeWare, Inc. and **IntelliCorp, Inc.** jointly announced that the merger agreement under which KnowledgeWare was to acquire IntelliCorp in a stock-for-stock exchange has been terminated. KnowledgeWare and IntelliCorp are continuing discussions with respect to possible cooperative development and/or marketing arrangements aimed at offering integrated CASE solutions for the UNIX marketplace.

Saber Software, Inc. has changed its name to **CenterLine Software, Inc.** It has also changed its products names from Saber-C to CodeCenter and from Saber-C++ to ObjectCenter. CenterLine has also announced cooperative marketing agreements with **Rational** and **Versant Object Technology** under which the companies will jointly market their products to C++ software development organizations. CenterLine also announced the integration of its ObjectCenter programming environment for C++ with Versant's ODBMS. CenterLine and Rational are exploring opportunities to produce a complete integration of ObjectCenter and Rational Rose.

ParcPlace Systems, Inc. and **Servio Corp.** announced that they will cooperatively integrate and market their products. Under the terms of the agreement, the two companies will cooperatively market and integrate Objectworks/C++ and GemStone to form an enterprise solution for C++ programmers.

The **Object Management Group Inc.** announced the adoption of the combined specification for the Object Request Broker from six companies—**Digital Equipment Corp., HyperDesk Corp., Hewlett-Packard Co., SunSoft, Inc., NCR Corp.,** and **Object Design, Inc.** The written specifications will be available in January directly from the OMG.

Object Design and **Spatial Technology, Inc.** have cooperatively integrated their products. ObjectStore is the only object-oriented database on the market fully integrated with Spatial Technology's ACIS geometric modeler.

ObjectStore has also been integrated with the STEP Tool Kit from **STEP Tools, Inc.** STEP Tools' products convert data from old legacy formats such as AutoCAD, DXF, or IGES files into STEP/PDES format.

Guidance Technologies, Inc. has signed a three year distribution agreement with ICL, a European-based information technology company. ICL has been given the nonexclusive right to use and resell Choreographer, GUIdance Technologies' graphic user interface development environment, through branch sales offices worldwide. ICL will also have exclusive distribution rights in Scandinavian countries. GUIdance Technologies also announced that it has been awarded a patent on its DisplayObject Editor, a key component of Choreographer. The new patent, No. 5,041,992, includes 17 claims that cover visual techniques for creating graphical screen elements.

IBM has selected **ONTOS, Inc.** as an Industry Application Specialist for the multimedia market. As an IAS, ONTOS becomes a partner with IBM in the multimedia market on a national basis. ONTOS also announced it will support OS/2 2.0. ONTOS DB currently supports OS/2 Version 1.3. Earlier, ONTOS joined the IBM Business Partner program as the first object database vendor.

Information Builders, Inc. and **Apple Computer, Inc.** announced a strategic relationship in which IBI's Enterprise Data Access/SQL client/server software will provide an Apple Macintosh personal computer access to a broad array of nonrelational data. The relationship will also include future software development by IBI for the Macintosh and joint sales and marketing efforts between the two companies.

Information Builders, Inc. announced a strategic alliance with **Soft-tek International, Inc.** that will provide high-resolution graphics capabilities to UNIX-based users of IBI's FOCUS.

Meridian Software Systems, Inc. and **Software Productivity Solutions, Inc.** agreed to form a strategic business alliance for the promotion and sale of SPS's object-oriented Classic-Ada software development tool set to the PC users market. Meridian will be selling the Classic-Ada product along with its own extensive line of CASE tools

Dr. David Maier, a scientist at the **Oregon Graduate Institute of Science** and **Technology,** will lead a national research team in designing new software and applications for object-oriented databases under grants from the **Defense Advanced Research Projects Agency** (DARPA) and the **National Science Foundation** (NSF).
Glockenspiel has been appointed as the exclusive European Authorized Marketing Agent for USL's C++ Release 3.0, a new product featuring the full implementation of "Templates." Glockenspiel, a Dublin-based software technology house, specializes in C++ for software development.

Digitalk announced support for **IBM's** OS/2 Presentation Manager in its release of Smalltalk/V PM 1.3. As part of IBM's support for Smalltalk/V PM, the company has licensed Digitalk to ship the dynamic link libraries for these new controls.

... continued from page 1

built by documenting the concrete flows on blueprints, but, alas, many business applications still are.

Even when we do use the tools of enterprise information modeling, we are dealing only with part of our problem. Enterprise data models enable the skilled analyst to represent static business data structures. But business rules are modeled—and implemented—separately from enterprise data, as are integrity and domain constraints. How is the dynamic behavior of data (which is entirely apart from application programs) implemented? Why is the data itself still disconnected from the meaning—the business definitions that make data something more than random numbers? These are the kinds of issues that object-technology supposedly addresses with a waive of the magic mouse.

Yet something is fundamentally wrong here. As our industry revels in its newfound truelove object-oriented everything, we even give our pet cats perverse orientations in the name of O-O. Once again, we are trying to solve the world's problems at the program code level.

Brad Cox makes what I think is the essential distinction for information professionals and for the data-drowned, computerized business of today: object-oriented programming is for developers; object technology is for people. Sounds so earthy, I think he's a closet Californian. It was Brad's concept of the "Software IC" that turned on the proverbial light bulb for me, triggering my transformation from a programmer to an ardent practitioner of engineering methods. Once again, his ability to put broad philosophy into a few words gives new meaning to our technology.

In light of Brad's observation, I think of object technology not in terms of programming technology. Try this on for size: objects are the things and concepts in the real world. For our businesses, products, customers, machines, employees, and bank accounts are objects. Object oriented is an attitude. Being O-O means focusing on real world things in contrast to the alternative, which I will call computer oriented. Object technology, then, is a conceptual tool for helping us focus on things in the real world.

What seems promising about object technology is the migration of computer technology away from the computer toward a focus on the real world. This technology could be used to raise the human condition from data drowned to, perhaps, information driven.

So what's "the right stuff?" How do objects fit? How can a business develop an integrated understanding of its shared data and shared behavior? How do business rules fit? And how does any of this help me with today's databases?

The insight given us by semantic information modeling is so obvious in its simplicity, yet limited in its paucity: healthy organizations must understand, manage, and share data. Wow, Mom, does this mean I have to share my toys? Yes, Robert, now why is that so hard? Well, if I have defined my power structure around information control, allowed my data to diverge from others' business knowledge, and defined my business facts from the narrow vantage of my department, it's hard!

Sound enterprise-level information modeling can dig a business out of this mess, but only part way. Entity modeling looks at stable structures very precisely, but allows no rules or dynamics. Sophisticated modelers have moved beyond isolated data to capture the business rules that give data its meaning. The best practitioners model at the enterprise level using definitions and precise entity naming (semantic modeling) until they obtain a consensus. Only then do they proceed to examine the attributes. In this way, information professionals can improve the business by catalyzing enterprise-wide change without getting lost in the corrupt data structures and turf wars of today. I didn't say this was easy, but those who make the effort to develop the social skills, business background, and rapport with business peers needed to gain credibility and influence in an organization really do this!

Brad Cox makes what I think is the essential distinction for information professionals and for the data-drowned, computerized business of today: object-oriented programming is for developers; object technology is for people.

But we're still not home free! Herculean efforts to share data by the same rules don't begin to address business function. Separately addressing shared data and private process make sense: reports run against shared data are a case in point. Perhaps we do share the reports, but maybe not. These aren't of paramount importance. The process that concerns me is the stuff that makes the business work. Where do the business rules get implemented so that business users (not programmers) are in control? What about rules that transform data from one state to another? What of the processes that drive work-flow systems—business process-control solutions that are dynamic emulations of human organizational behavior in which the computer becomes a real-life player instead of a passive data server? We still don't hear about shared

process above the level of shared functions in edit libraries. While data sharing concepts have become relatively sophisticated, process sharing is still viewed as a programming-level problem. By analogy to Brad Cox's observation, data sharing is at the object technology level; shared process is still at the OOPS level. This, I would suggest, is where object technology can help.

Object technology at the enterprise level is about understanding the strategic things of the business. These things are usually shared among different business actors in divergent ways, at different points in time. There are rules in the sharing game that the things need to enforce because we actors won't. We actors know something about the things; there is also information that

> ***Enterprise object modeling is a superset of enterprise information modeling: data elements and stable structures (relationships), plus behavior, rules, and temporal transition.***

the things know about themselves to which we actors are not privy. As the things take on rule enforcement, they adopt behavior. When behavior is more than simple integrity enforcement, we're talking about shared process—instilling the smarts of our business into the things that we use, manipulate, sell, track, etc. A logical extension of this is to give the behavior intelligence through embedded expert systems. Ultimately, the distinction between things and actors is the stuff of philosophy: we maintain an illusion of difference to support our own self-esteem, but we're creating objects in our own image. Far out?

Now replace "things" with "objects." For the implementation minded, we've walked around the definition of objects that has been central to OOPS for several years; only the stakes have been raised from the program to the enterprise. Embedding expert systems capabilities in methods (the OOPS term for behavior) and using methods to enforce business rules is nothing new. What is new is thinking about the business in terms of objects. Not computer structures—real-world structures. Today's approach to thinking about shared data in terms of entities is a step removed from talking about tables. Object technology can move us a level of abstraction further away from the implementation (and OOPS, pleasantly, can move implementation a level of abstraction further away from the computer, as well). In short, object technology is defining computer reality in terms of business reality. Computer-oriented technology admirably attempts the inverse.

Enterprise object modeling is a superset of enterprise information modeling: data elements and stable structures (relationships), plus behavior, rules, and temporal transition. Instead of packaging data into entities, processes into programs, and stringing the meaning and rules somewhere between the two, we package the whole caboodle into business objects. The object knows certain things about itself (that, presumably, the business user is interested in knowing); has the ability to maintain its own validity by enforcing business rules; and can participate—again, by the business rules—in behavior that involves other business objects. Object technology also provides a mechanism that allows business users to agree to disagree through the concepts of inheritance and overloading. Inheritance allows a business object to be defined in terms of another business object; overloading is one way of allowing it to be different. In many respects, this concept of enterprise object modeling addresses in an organized package the things we need to know but lack the tools to represent. As Marie Lenzi explains it, object technology is sort of WYGIWYS (that's what you see is what you get, with a backward orientation!).

Clearly, the maturation of object technology can be used to our advantage. While it won't stem the tide of useless information, it will give us as managers and information professionals a toolkit for correcting the problem—if and only if we are willing to do it ourselves. This article raises some of the issues that need more thought—some new, some building on the known. These are among the things that future issues of the Hotline will address. Regardless of your leanings on object technology, remember the admonition found on every new automobile sold in the US: "Warning: Objects in the mirror are closer than they seem." ≡≡

IDB introductory package
Persistent Data Systems, Inc.
75 West Chapel Ridge Rd.
Pittsburgh, PA 15238
(412) 963-1843

Persistent Data Systems, Inc. announced the availability of a new introductory package for its IDB object database, specially priced at $99. **The IDB introductory package** includes a working application that runs under the IDB browser, a 150-page tutorial, the database schema and C source for the working application, and a $99 credit toward the purchase of the full IDB development system. Versions of the introductory package are available for Windows 3 and the Macintosh. The tutorial assumes that the reader knows how to program in C, but does not require any prior experience with object-oriented systems. The Persistent Data Systems' IDB object database is a fully object-oriented database that runs on wide range of personal computers and workstations including Windows 3, Macintosh, NeXT, HP (both Domain/OS and HP/UX), and Sun. IDB can be configured to run either stand-alone or networked with any mix of supported platform using a peer-to-peer architecture.

Object-oriented database manipulation system
VC Software Construction GmbH
Petritorwall 28
3300 Braunschweig
Germany 49531 24 24 00

VC Software Construction has launched a new **object-oriented database manipulation system**. The ODBMS has been implemented by using Smalltalk/V to support the software platforms OS/21.x Presentation Manager and Windows 3.0. Its storage facilities of objects can be used during the development of Smalltalk applications as well as by a stand-alone database application. ODBMS will be delivered with an integrated graphical user interface. It stores arbitrary objects such as data, text, and images in opposition to relational databases. The integrated manipulation system of ODBMS allows the creation of databases that store objects with arbitrary attributes.

Matrix.h++/ Linpack.h++
Rogue Wave Software, Inc.
P.O. Box 2328
Corvallis, OR 97339
(503) 754-2311

Rogue Wave Software, Inc. has started shipping two new C++ class libraries, **Matrix.h++** and **Linpack.h++**. These new C++ class libraries extend the C++ language to include numerical algorithms that were previously available only in Fortran. Both Matrix.h++ and Linpack.h++ are completely compatible with Rogue Wave's other C++ class libraries, Tools.h++ and Math.h++. Matrix.h++ and Linpack.h++ are available for most machines (from MS-DOS to UNIX) including a vectorized version for the CRAY. Matrix.h++ includes all the functionality of Math.h++ and specialized matrix classes such as banded symmetric, positive-definite, Hermitian, triagonal, etc. Linpack.h++ includes all of Matrix.h++ plus all of the functionality in the Fortran version. However, the traditional messiness of the Fortran version has been replaced with high-level, yet efficient, objects that make code easier to write and maintain.

online Smalltalk market
The Agorics Project
10364 Bridgetown Pl.
Burke, VA 22015
(703) 250-4760

The Agorics Project has announced the opening of an **online Smalltalk market** on the AMIX electronic marketplace. AMIX is provided by Autodesk, a subsidiary of the American Information Exchange Corporation. Smalltalk developers will be able to buy and sell classes, methods, tools, frameworks, and other Smalltalk-related information. Features include electronic mail, negotiation facilities, hypertext linking and navigation, customizable personal "notebooks" to facilitate record keeping, and central billing and payments by AMIX.

Calendar Object
Serius Corp.
1981 E. 4000 South
Salt Lake City, UT 84117
(801) 777-7788
(800) 876-6847

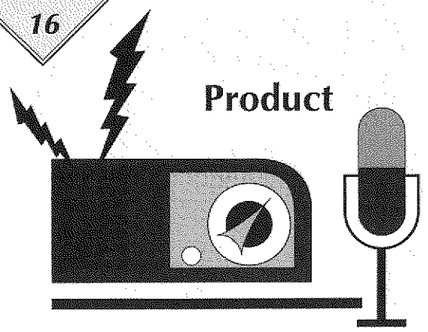
Serius Corp. has introduced the Serius **Calendar Object**, which allows users of the Serius Programmer and Serius Developer desktop programming environment for the Macintosh to incorporate calendars into the applications they build. The Calendar Object can be used to create appointment calendar applications complete with alarms to alert end users to upcoming events. Calendars can be customized by changing their size, font type, color, and layout. In combination with Apple Events, the Calendar Object can create appointment calendars that communicate over a network. The Calendar Object is part of the Serius Object Library, a library of 46 prefabricated components that users can combine in numerous ways to create custom applications.

Star Sapphire Common LISP Version 3.2
Sapiens Software Corp.
P.O. Box 3365
Santa Cruz, CA 95063
(408) 458-1990

Sapiens Software Corp. has released **Version 3.2** of its **Star Sapphire Common LISP** for the PC, PS/2s, and compatibles. The new version includes over 80% of the Common LISP specification. New features include the use of virtual memory in the editor, improved support for LISP compound types, conformance with proposed ANSI Common LISP, and an interactive text screen designer. Also announced was a new pricing policy for academic copies of Star Sapphire: $50 per single copy with proof of academic affiliation and a volume price of $40 in quantities of 10 or more.

Personna
InnoVision Software, Inc.
309 Pinevue Dr.
Monroeville, PA
(412) 392-1580

Innovision Software has announced the released of **Persona**, their latest Actor add-on product. Persona provides a full-featured programmer's editor for use within Actor. It consists of a set of class libraries that, when loaded into Actor, add many powerful editing features to the Actor browser. Some of these features include multiple windows, multiple method buffers, and undo/redo capabilities. Persona also provides an excellent collection of popular programming editor emulators. Persona is currently available for Actor Versions 3.0 and 4.0. BRIEF and Epsilon emulators are included with the product. Additional emulators will also be available.



Product Announcements

Product Announcements is a service to the readers of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY; it is neither a recommendation nor an endorsement of any product discussed.

Object Center Version 1.1
CenterLine Software, Inc.
10 Fawcett St.
Cambridge, MA 02138-1110
(617) 498-3000

CenterLine Software, Inc. (formerly Saber Software) announced a major new release of ObjectCenter (formerly Saber-C++). **ObjectCenter Version 1.1** incorporates new capabilities, better support for large projects, and performance enhancements requested by users deploying C++ in critical development projects. By supporting shared libraries on the Sun platform, ObjectCenter users can now load and link functions and data from shared libraries on Sun-3 and Sun-4/SPARC platforms. ObjectCenter 1.1 fully supports the industry standard cfront 2.1 implementation of C++. ObjectCenter is a dual C and C++ language programming environment that integrates into one comprehensive user extensible source-level debugging, a suite of graphical browsers, an incremental linker, and C++ error-detection system.

PC/Focus Release 6.0
Information Builders, Inc.
1250 Broadway
New York, NY 10001
(212) 736-4433

Information Builders, Inc. introduced the new **PC/FOCUS Release 6.0** for the IBM PC with enhanced database development features and completely redesigned menus for a more graphic orientation. PC/FOCUS is menu driven with a graphic point-and-click interface. Features include a job scheduler and browser. The new PC/FOCUS 6.0 is also able to access IBI's Enterprise Data Access (EDA)/SQL family of client/server products which provide transparent access to more than 45 different data file types on over 35 different operating system platforms.

Eiffel/ArchiText
Interactive Software Engineering, Inc.
270 Storke Rd., Ste. 7
Goleta, CA 93117
(805) 685-1006

Eiffel has been ported to HP 9000/400 and HP 700 machines. Eiffel is an object-oriented language and environment for software engineering. It is aimed at developers concerned about the quality of the software they produce—particularly reusability, extendibility, and efficiency. Now that Eiffel has been ported to HP, the current Version 2.3 is available on all major UNIX platforms including Data General, DEC, IBM, MIPS, NeXT, and SUN. DOS and Macintosh versions are currently being developed by companies in Germany and the US.

ArchiText, the language-driven document constructor designed to write and maintain structured documents, is now available for Smalltalk and C/C++. Developed by Interactive Software Engineering, ArchiText uses purely object-oriented programming techniques and is written entirely in the Eiffel object-oriented language. ArchiText is a powerful high-level tool for manipulating and viewing structured documents, such as programs, specifications, and technical or administrative reports. A unique feature of ArchiText is its ease of adaptation to any context-free language. It features a graphical user interface based on the Motif GUI standard and caters to the needs of both beginning and advanced users of a programming language or other notation.

UniSQL/X UniSQL/4GE Application Development Environment
UniSQL, Inc.
9390 Research II, Ste. 220
Austin, TX 78759-6544
(512) 343-7297

UniSQL announced their unified relational and object-oriented database management system, **UniSQL/X,** for Sun SPARC workstation users. UniSQL/X formally extends the relational data model and standard ANSI SQL query language with support for arbitrary data types, methods, multiple inheritance, encapsulations, and complex multimedia data integration.
UniSQL also announced the **UniSQL/4GE Application Development Environment** designed to help UniSQL/X users build applications that take advantage of object-oriented technology and multimedia data integration. UniSQL Technology Partner Program and UniSQL Early Customer Program participants will receive January delivery. Both products will become generally available during March.

Xmath
Integrated Systems, Inc.
3260 Jay St.
Santa Clara, CA 95054-3309
(408) 980-1500

Integrated Systems, Inc. has begun shipping **Xmath**, an object-oriented mathematical analysis and scripting environment designed specifically for the X Window System environment. Xmath represents a new generation of numerical computing tools that incorporate application-specific engineering objects (ASEOs) that improve computing speed, accuracy, and ease of use. ASEOs bring the power of object-oriented programming to engineers and scientists. This is comparable in value and use to a writer's word processor or a financial manager's spreadsheet.

Send Product Announcements to Marie A. Lenzi, 155 East 31st, #22S, New York, NY 10016. Fax (212)725-0224. Include company name, address, and phone number.

GETTING STARTED

Adopting objects: pitfalls

by J. Michael Connell

Last month, we introduced the process for evaluating and adopting the technology of objects. That article presented a model for the process, and this column will identify weakness and suggest ways to avoid some of the most common mistakes.

THE TECHNOLOGY ADOPTION PROCESS REVIEWED PROOF-OF-CONCEPT

The initial phase of the technology adoption process is the exploration phase, which has two stages—prospecting and proof-of-concept. The exploration phase is conducted by the gatekeeper labs. Their role is to locate, study, characterize, and monitor any and all new technologies that show a potential for becoming important to the company. Typically, the gatekeeper lab undertakes the earliest studies on objects.

The purpose of the first stage, the prospecting stage, is to acquire sufficient knowledge about objects to determine if they are real and might have value to the company. The purpose of the second stage, the proof-of-concept stage, is to determine if the objects truly do deliver on the promises of their proponents. Ideally, this project would also demonstrate the value of the objects in the context of the company's business.

The middle phase of the technology adoption process, the technology transfer phase, contains stages 3 and 4—evaluation of alternatives and pilot trial. This phase has three purposes: to evaluate objects more thoroughly in light of the company's computing needs; to transfer the knowledge, expertise, and know-how of objects from the gatekeeper lab to a core group of application developers; and to use objects in a medium-scale project to evaluate this technology in a production environment.

The deployment phase completes the technology adoption process by introducing objects into the mainstream of the company business. The initial production stage implements a strategically important application to verify robustness, resiliency, and reliability. The diffusion stage deploys the technology of objects throughout the company by building success upon success.

STORIES OF DIFFICULTIES AND TRIUMPHS

The following stories are loosely adapted from real experiences collected from companies that are conducting critical evaluations of objects. These firms include a financial institution, a transportation company, an industrial products company, a computer services company, a software products company, an energy company, and a computer company. The looseness of the adaptation is deliberate in order to hold in confidence the names of these firms while disseminating some of the lessons they have learned. I am certain that many readers will see similarities between these experiences and the experiences they may have had in their own firms.

Inadequate charter for the gatekeeper lab

Far and away the most common problem encountered during the technology adoption process is improper management of the gatekeeper function. The gatekeeper lab is generally considered an R&D function. The charter of this lab is to maintain thorough comprehension of emerging technologies and, because of this view, it often develops an external focus. It is staffed with top technologists, and these technologists become champions for their favorite emerging technology. Some of the draw-

J. Michael Connell is a consultant in Portland, OR specializing in the marketing and management of object-oriented products, tools, and environments. During most of the formative years of object-oriented systems, he has been in roles key to the development, market positioning, and penetration of this technology.

backs experienced with these groups are that gatekeeper staff may not be closely connected with internal developers and can become somewhat remote from company business (ivory tower syndrome) or that the staff are allowed to be self-managing with few demands and fewer measures of performance (loose cannon syndrome).

At one firm, the gatekeeper lab did not see, as part of its charter, the requirement to sell objects to the operating groups. They considered themselves the repository of knowledge about new technologies rather than a conduit for that knowledge. In that organization, it was left to an operating group to approach the gatekeeper lab to seek assistance in evaluating objects. This attitude is all too prevalent and counterproductive

> ***We all know examples of good technology that never made it into a company's operations but went on to enhance the competition.***

The charter of any gatekeeper lab must unambiguously identify its role as a service organization to the development groups within the operational units. The effectiveness of this organization must be judged from the prospective of the operational groups. A popular way for forcing a gatekeeper lab to address the needs of the operational groups is to require that most of its functions be supplied directly from the budgets of these groups. I endorse the approach where a subsistence level of funding is guaranteed by the corporation and all additional funding comes as a result of selling services to the operational groups. I also recommend that the staff of a gatekeeper lab include a salesman who understands the selling process and uses the same techniques for internal clients that would be used to sell these same services to external clients.

Stuck in the exploration phase with no way out

It is not uncommon for proof-of-concept projects to remain active in the gatekeeper lab for long periods of time without being successfully transferred into the operational side of the company. When this happens, it is an indicator that management has not given adequate consideration toward developing a mechanism for transferring objects into an operational group or coming to closure within the lab.

Proof-of-concept projects should be limited in scope and time and should have a predetermined set of goals. These goals should include a mechanism for effecting an orderly transfer of objects to an operational group. This mechanism will differ from company to company and maybe even from project to project within a company. It is important that all the proof-of-concept goals be achieved in a reasonable time frame; otherwise, the project should be scrubbed and the technology returned to the exploration stage.

Timing must be considered

During the proof-of-concept stage, the rate of change in the technology should also be considered. If new concepts are emerging at a rapid pace and these new ideas supersede older concepts, then it may well be prudent to wait until the rate of change slows down before considering moving beyond this stage. A pause here may avoid the danger of adopting a technology that will become obsolete in a very short time. In the case of object's, though, change appears to be continuing but at a steady and tractable pace and should not present undue risk.

Dropping the ball in transfer

The technology transfer phase is the most often mishandled part of the overall technology adoption process. We all know examples of good technology that never made it into a company's operations but went on to enhance the competition. The primary difficulties in this transfer process generally stem from misunderstandings that occur between the gatekeeper function and the applications development groups in the operational areas.

A manager in an applications development group will not endorse even the trial use of objects until he is convinced that they present a potential benefit for him. He must see clearly that objects can offer a solution to one or more of his problems or, better yet, one or more of his most serious problems. And, he must be convinced that objects represent the best solution to these problems. Also, he must be assured that the use of objects will not result in any new problems. The management of the gatekeeper lab and the technical champions for objects must be prepared to help the applications development manager answer all these questions.

This "internal education" activity is one of the most important steps of the technology transfer phase and is often neglected. This task is time consuming and one that many technologists abhor; therefore, it often goes undone. One company that organized a gatekeeper lab specifically for object technology has learned—after more than a year—that educating the internal operational groups as to the value of objects and the relevance to their needs was the key to their success.

Industry News

develop for the platform. "The big thing that NeXT has over anyone else is that it is cheaper to develop for NeXTStep than anything else," said Marc Munford, marketing manager for Insignia Solutions, Inc....

The Next Step: Jobs Outlines His Strategy, Brian Cillooly & Heather Clancy, ***Computer Reseller News****, 10/4/91*

...Microsoft, however, cannot see a fit right now for the [object request] broker in its OLE strategy. "We're looking at the broker as an emerging standard in the UNIX community," said Edward Jung, a Microsoft software architect. "It's not so clear that it's useful running on personal computers. It's a little less optimized for our applications." As specified now, the broker occupies too much workstation memory, Jung said. That is less of a concern for UNIX environments than it is for DOS and Windows environments, he added. Microsoft is also developing code internally to optimize OLE for networks, Jung said. It has to evaluate whether there is a role for the broker in that code. Nonetheless, Microsoft is trying to determine if it can strip out and implement pieces of the software and still maintain OMG compliance and interoperability with the broker, Jung said...

Firms Mull Support for OMG Task Broker, Jim Duffy, ***Network World****, 11/11/91*

...While integrators mostly applauded the Object Request Broker specification, there was skepticism about its usefulness without a single specification for what an object actually looks like within a distributed environment. The specification work to define an object model will get under way in 1992, according to the Object Management Group. Leon Smith, president of Paradise Software group, an object technology consultancy based in Paradise, CA, would have preferred to see the object model specified first. "I am disturbed by the order they have taken [in the standards process]," said Smith. "The definition of the object is critical before you specify the messaging protocol"...

Object Group Releases Spec, Joshua Greenbaum, ***Computer Systems News****, 11/4/91*

...About six weeks ago, [IBM] quietly added System Object Model (SOM) technology to OS/2 2.0, which is basic object technology developed by IBM that will carry over to the Taligent operating system jointly developed by Apple Computer Inc. and IBM. Joseph Guglielmi...and John Soyring...[both of IBM] last week confirmed that SOM had been added to OS/2 2.0 and would carry over to the vendor's work with Apple. Both noted that IBM hoped to clarify to its customers that OS/2 2.0 would retain compatibility with their forthcoming Taligent system software being co-developed with Apple and would be supported by that operating system, similar to how OS/2 2.0 currently supports Windows and DOS...

OS/2 Battle Heats Up at Comdex, Wendy Goldman Rohm, ***Computer Reseller News****, 10/28/91*

...The growing use of such techniques as Formal Methods and Object-Oriented Design should lead to productivity improvements and significant quality enhancements. The common factor that these and other new techniques have is that they are founded on the basic philosophies that structure will defeat complexity and that prevention is better than cure.

Software Quality: A Pragmatic Perspective, E. Reed Doke & Thomas F. Buckley, ***Information Executive,*** *Fall 1991*

Languages & Tools

...Since Borland C++ is the first Borland C product that lets you develop Windows programs, it has gotten the reputation for being strictly a Windows development tool. Not so. If anything, Turbo Vision, the profiler, and the stand-alone debugger make Borland C++ the best DOS development system around. Even if you hate Windows with a passion, and never intend to use the Windows compatibility, the remainder of the product's features make Borland C++ a compelling purchase...

Borland: From C to Shining C, Nicholas Petreley, ***Trenton Times****, 10/7/91*

...The truth is that ObjectVision is a powerful tool for developing and operating data entry screens using a graphical environment, but that is all. ObjectVision cannot select records, query or sort a file, or generate a report. The limits of its print facility are an embarrassment: It can print only complete data entry forms and the decision tree logic that underlies calculation among fields. It can't generate a simple list of selected field entries. How can Borland call ObjectVision the foundation of an interactive business application when the words "sort," "query," and "report" don't even appear in the reference manual's index? Instead, ObjectVision is a general-purpose, Windows-based form generator that employs the now common object-oriented programming...There are several quirks in the program. For example, you cannot select a printer from a drop-down menu but must leave ObjectVision and select it from the Windows control panel. Also, there is no keyboard equivalent for the Next and Previous Record mouse buttons, nor are there corresponding options on the drop-down menus, so you can't scan records without a mouse. My suggestion: Wait until Release 2.0 before buying this product...

User World: Borland's ObjectVision Stresses Gimmickry Over Capability, Jack Stone, ***Federal Computer Week****, 10/21/91*

FYI

Excerpts from leading industry publications on aspects of object technology

Software Configuration Management

...In traditional SCMs, the disk file is the primary data type that gets managed. The basic properties of the disk file are its name, size, location, and content. By contrast, an object-oriented SCM uses a range of data types with properties such as program functionality, hardware dependencies, and foreign languages support. The advantage here is that object properties have meaning outside the physical definitions of the software. File-oriented methods can only say where the physical chunks of a program reside. An object-oriented system can also say where the functions of a program reside and which of those functions satisfy specified criteria—such as which changes go with which hardware platform or program feature...An advantage of organizing information as objects, rather than as source files, is that no inherent restriction exists regarding the type of information represented. Differences in machine-readable files can be handled as readily as differences in text files...

Object-Oriented Software Configuration Management, Richard Harter, ***Dr. Dobb's Journal****, 10/91*

Adopting Object Technology

...When I learned about object-oriented programming using Smalltalk, I was preoccupied by the picayune details of the language. I spent the first six months understanding the subtleties of the syntax, learning the class libraries, studying the language semantics and implementations, and mastering the programming environment. At the end of that first half year, I had a solid grasp on all the little issues of an object language, but I still knew nothing about objects. I had been reimmersing myself in issues familiar to me from my days as a procedural programmer. I focused on the non-object-oriented aspects of my object-oriented language to avoid the uncomfortable feeling that I didn't know what was going on. By clinging to my old ways of thinking, like a nervous swimmer to the side of the pool, I was preventing myself from reinventing my perspective. It was only through patient and expert tutelage that I was able to break free of my old habits and begin to make use of the power in objects. I now know that learning objects needn't be frightening or confusing...If you're willing to trust yourself to learn the syntax and programming environment later (after all, you've probably learned several of each already), you can be doing objects in a few hours, too...

Think Like an Object, Kent Beck, ***UNIX Review****, 10/91*

...Let's assume...that you decide an [Object Based System] would be sufficiently faster, sufficiently easier to maintain, sufficiently more reliable, and sufficiently more extensible to be worth the cost. How do you proceed? Very cautiously. The manager I talked with suggested three major steps:

- Review your current in-house software and remove non-essential or useless functions before redesigning it with object tools.
- Sort out all the dead code from the old system.
- Rethink all implementation issues in the new design, especially the interface.

To which I will also add these thoughts: First, train your developers, letting them cut their teeth on small systems before they tackle the conversion. Second, expect their current productivity to take a nose dive. Above all, remember that in-house software tends to have a very long life. The number of viable RPG II and COBOL applications out there stand as silent testimony.

The Mac Manager: Obscure Objects of Desire—Dealing with OOP, OBS, Don Crabb, ***MacWeek,*** *10/29/91*

...[Microsoft president Mike Hallman:] "There is no point in time where we suddenly go from the nonobject environment to the object-oriented involvement. We'll evolve over time as the technology becomes refined and available and, hopefully, we will be able to accommodate the world that's already there, just like the Windows world accommodates DOS character applications. You want to be able to have old and new run together"...

Microsoft Talks MIS, David R. Brousell, ***Datamation,*** *10/15/91*

Industry News

"Hel...is a place where nothing connects with nothing." With that T.S. Eliot quote looming large on the screen behind him, the U.S. Department of Defense's top information systems official underscored the hunger for interoperability in today's corporate computing environment. An equal helping of honesty about the confusion surrounding open systems is also needed, said Paul Strassmann, director of defense information for the DOD, in his keynote speech last week to X/Open's World Congress on Open Systems. "It is very important in the world of open systems that we don't pretend to know all the answers," Strassmann said. "Being explicit about the risks involved will increase your credibility with your senior managers"...Reuse of software code and components will be a "principal contributor" to the value of open systems, Strassmann said. "This is a set of benefits you can explain to top management." Strassmann also urged IS executives to develop long-range open systems strategies for their companies, emphasizing economic value rather that technical capabilities. "The primary inhibitors to open system will be managerial, after the technical problems are solved," he said...Indeed, many independent software vendors cite NeXTStep's object-oriented approach to applications building as their main reason for choosing to

Throw it over the wall but no one home

Not all the communications problems in the technology transfer phase lie with the gatekeeper lab. In the vernacular, this transition is sometime known as "throwing it over the wall." For this to be successful there must be a "thrower" and a "catcher." Unfortunately, all too often there is no one on the operational side prepared to catch the new technology and run with it.

The most common reason an operational group will give for this lack of reception is that everyone is so busy dealing with day-to-day problems that they don't have the time or the funds to give serious consideration to something new. If this is a persistent state, then it is highly unlikely that any new technology will ever reach full adoption. There are examples of gatekeeper labs whose effectiveness was very low simply because they could never get sufficient attention from operational groups to make an orderly transfer of any technology.

The secret to a successful technology transfer is to make this transition a joint effort, beginning in the proof-of-concept stage. The needs of one or more of the operational groups should be identified and the proof-of-concept project should be tailored to address those needs. An engineer from the operational group should join the proof-of-concept project team in a full-time, active role. As the undertaking advances from the proof-of-concept stage through alternative evaluation and pilot trial stages, it should remain a joint project. In these succeeding stages, the operational group should assume an increasing portion of the effort. However, the champion from the gatekeeper lab should remain actively involved in a mentor role through completion of the pilot trial.

The right solution for the wrong problem

As stated above, an operational group must see objects as a solution to one or more of their problems. Occasionally, even though a technology has terrific benefits, it may not be relevant to the immediate needs of any internal user group. This may occur as a result of poor selection of candidate technologies in the initial stage. However, even in the best of circumstances, some technologies may fail to be accepted at first. When this occurs, it is important to recognize that the technology adoption process is stalled and to assess the reasons for this blockage. The most common reason is the customer's problems were poorly understood from the onset or may have changed during the adoption process.

Technology adoption by management edict

One of the most dangerous problems faced by some companies is the propensity of some corporate managers to edict the change to object technology based solely on early exploratory results. Strong technical champions that have the ear of corporate management can sometimes make a convincing case for skipping the middle stages and jumping directly from stage 1 or 2 to stage 5 or even 6.

A senior vice president told me that they had done literally dozens of successful proof-of-concept projects but they did not know what to do next. They were considering a corporate management edict to establish a large development group dedicated to objects and jump immediately to stage 5.

In another case, the CEO of a firm, himself a former engineer, made the strategic decision to convert their entire applications development to the use of objects. His belief was that objects are an important technology that would provide a competitive lever to the first firm in his marketplace to adopt them. His rationale may very well prove to be correct in the long run, but his firm has experienced many difficulties and delays as a result of this dictate. I suspect that his timing was wrong and suggest that the orderly process of a normal technology adoption process would have indicated this error and saved his firm untold trauma without risking any loss of his competitive position.

In yet another case, a firm jumped directly into stage 5, initial production, with no previous exposure to objects. This was accomplished by relying heavily on consultants for planning, design, and implementation. These consultants all had previous experience in stages 1 through 4. This project will succeed. And, even though it has been expensive, it may yet be demonstrated to have been cost effective.

DOING IT RIGHT

So far I have found only one firm that I believe has been entirely correct in their approach to the evaluation and adoption of objects. They have not yet completed all six stages. However, they have made solid and orderly progress through stages 1 through 3 and are now engaged in stage 4. The secret to their success is taking the time to do it right. They have been engaged in this endeavor for over three years, and I expect that they will continue for two more before they have completed the initial production stage.

Objects are a breakthrough technology that have the potential to make radical impacts on the effectiveness of tomorrow's applications development environment. But, as is often the case with revolutionary ideas, adopting object can be a double-edged sword. Proper methodology, careful planning, and patient execution are the keys to success. There will certainly be some companies that will try objects and reject them. Unfortunately, most of these rejections will be the result of an inadequate methodology for adopting technology and will have very little to do with the value and the appropriateness of objects. ≡≡

OBJECTS DOWN UNDER ≡≡

Adoption rate of object technology: a survey of NSW industry

by B. Henderson-Sellers

Recently, the development of object-oriented systems has become a mainstream topic in academic, technical, and professional literature.

There are now many articles that cover topics such as object-oriented methodologies,[1] systems notation,[2] project management,[3] development experience,[4] and object-oriented languages.[5,6,7] With this flurry of activity in the literature, one may be forgiven for thinking object-oriented techniques now make up the standard design methodology of many organisations, replacing other more traditional and well-known methodologies, including structured design,[8] information engineering,[9] and Jackson structured design.[10] On the other hand, it may be suggested that adoption by the information systems (IS) departments of commercial organisations of object-oriented analysis and design methodologies has not been as rapid as might have been anticipated some years ago.[11,12,13] One problem in obtaining clear information on the subject is that those organisations committed to the introduction of object-oriented techniques see this as giving up their competitive advantage. Consequently, the knowledge that they are employing these leading-edge, emerging technologies is frequently regarded as a "commercial secret." Thus, at present there are no firm figures on the adoption rate of object-oriented methods by the software industry, and there is only anecdotal evidence on the method of adoption. The broad aim of this research was to investigate the adoption of object-oriented methods within the software industry of Australia and to provide some solid evidence on how the industry is responding to this new approach to systems development. Therefore, with the cooperation of Sydney-based industries, we have undertaken a survey (in late 1990) to try to evaluate, without loss to the firms of commercial confidentiality, the overall perception the information systems industry has of the object-oriented philosophy.

B. Henderson-Sellers and J. M. Edwards are with the School of Information Systems, University of New South Wales. They can be reached at brianhs@usage.csd.unsw.oz.au and juliane@usage.csd.unsw.oz.a, respectively.

QUESTIONNAIRE DESIGN

The motivation for this research was twofold: first, to understand how organisations view the object-oriented paradigm so that future research could be targeted more effectively to their needs and, second, to attempt to provide educators with information that might be usefully employed in future educational course designs. Many industrialists are eager to note the need for adequate educational facilities, both in terms of traditional courses and also short, intensive, professional development courses for the object-oriented paradigm to be both understood and implemented at executive and technical levels.

In this survey, we have concentrated upon the adoption rates of object-oriented design and analysis techniques and attempted to discover whether published methodological approaches are being adopted, whether in-house development of methodologies is being undertaken, or whether an ad hoc approach is more prevalent. In addition, we have sought to evaluate the types of sources of information used in such a learning curve.

We were particularly interested in whether industries were adopting object-oriented ideas totally or incrementally at one of the three broad phases of the software engineering lifecycle (analysis, design, and coding). Possible methodologies for the "pure" object-oriented lifecycle are discussed by Henderson-Sellers and Edwards[14] and more pragmatic discussions on "hybrid" technologies that incorporate part functional decomposition/structured design, part object-oriented techniques, can be found.[15,16]

The survey briefly addressed the topic of languages and techniques used by industry in their methodologies. We were not so much interested in the specifics of any one methodology, but more in the way object-oriented

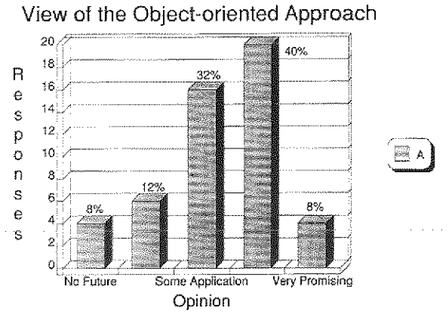


Figure 7. Respondents' perception of the likely future for object technology. Categories range from pessimistic on the left (no future) to optimistic (very promising) on the right. (The vertical scale is absolute number of respondents from the sample.)

the enthusiasm for obtaining information on object-oriented techniques in the Sydney area is extremely high and firms are sponsoring members of their organisations (both technical and managerial) to attend educational training courses and become members of the special interest groups of the Australian Computer Society, the adoption of object-oriented methodologies in any formal sense is essentially slow. Lack of published methodologies (as of the date of the survey) by "big names" seems to be a real factor in this slow growth and is compounded by the reticence of some hardware vendors to endorse the paradigm and provide software support for the new language environments available. The outlook for object-oriented techniques, however, remains bright, with methodologies, software tools, and professional courses all currently being developed to support commercial MIS departments that decide to move toward the object-oriented paradigm. ≡≡

Acknowledgments

We are grateful to the many MIS managers and software developers, especially those in the Sydney CBD, who participated in this survey. We also wish to acknowledge financial support for this project from the University of New South Wales Faculty of Commerce and Economics Special Research Fund.

References

1. Wirfs-Brock, R., B. Wilkerson, and L. Wiener. *Designing Object-Oriented Software*, Prentice Hall, Englewood Cliffs, NJ, 1990.
2. Wasserman. A., P. Pincher, and R. Muller. An object-oriented structured design method for code generation, *ACM SIGSOFT, Software Engineering Notes*, **14**(1), 32–53, 1989.
3. Thomsett, R. Management implications of object-oriented development, *ACS Newsletter*, October, 5–12, 1990.
4. Waldo, J. OO Benefits of Pascal to C++ Conversion, *The C++ Report*, **2**(8), 1–5, 1990.
5. Meyer, B. *Object-Oriented Software Construction*, Prentice Hall, Englewood Cliffs, NJ, 1988.
6. Cox, B. J. *Object-Oriented Programming*, Addison-Wesley, Reading, MA, 1986.
7. Weiner, R., S. Richard, and L. Pinson. *Introduction to Object-Oriented Programming and C++*, 1988.
8. Yourdon, E., and L. L. Constantine. *Structured Design*, Prentice Hall, Englewood Cliffs, NJ, 1979.
9. Martin, J., and C. Finkelstein. *Information Engineering*, Prentice Hall, Englewood Cliffs, NJ, 1981.
10. Cameron, J. An Overview of JSD, *IEEE Transactions on Software Engineering*, **12**(2), 222–240, 1986.
11. Howard, G. S. Object-oriented programming explained, *Journal of Systems Management*, **39**, 13–19, 1988.
12. Meyer, B. From structured programming to object-oriented design: the road to Eiffel, *Structured Programming*, **1**, 19–39, 1989.
13. Burgess Yakemovic, K. C. The bottom line: using OOP in the commercial environment, *Workshop Report, OOPSLA '89 Workshop*, 1990.
14. Henderson-Sellers, B., and J. M. Edwards. The object-oriented systems life cycle, *Communications of the ACM*, **33**(9), 1990.
15. Constantine, L. L. Objects, functions, and extensibility, *Computer Language*, **7**, 34–56, 1990.
16. Henderson-Sellers, B. Hybrid object-oriented/functional decomposition methodologies for the software engineering life cycle, *Hotline on Object-Oriented Technology*, **2**(7), 1–8, 1991.
17. Meyer, B. Tools for the new culture: lessons from the design of the Eiffel libraries, *Communications of the ACM*, **33**(9), 1990.
18. Barber, G. R. The Object Management Group, *Hotline on Object-Oriented Technology*, **2**(5), 17–19, 1991.
19. Booch, G. *Object-Oriented Design with Applications*, Benjamin/Cummings, Menlo Park, CA, 1990.
20. Coad, P., and E. Yourdon. *Object-Oriented Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1990.
21. Henderson-Sellers, B. *A Book of Object-Oriented Knowledge*, Prentice Hall, Sydney, 1991.

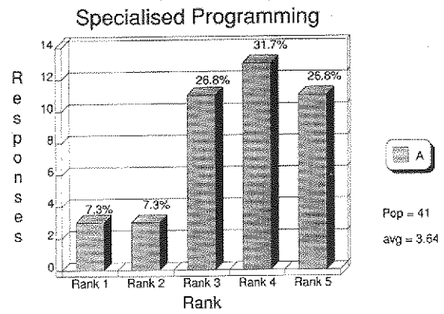


Figure 5. Percentages of respondents who placed the need for specialised programming courses in each of the five ranking positions.

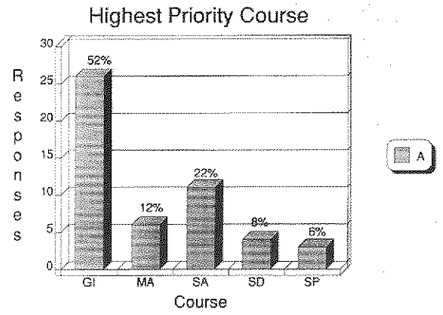


Figure 6. Percentages of respondents who ranked each of the five proffered course types as their first priority. Here, GI stands for general introductory; MA for more advanced; SA for specialised analysis; SD for specialised design; and SP for specialised programming.

oriented techniques are now being seen as early lifecycle methods as well as implementation techniques. The demand by industry for analysis and design courses reflects the growing number of articles, methodologies, and debates that are concerned with the application of object-oriented methods to the whole development lifecycle, recognising the maturing nature of the approach.

Projects

Direct questioning regarding the number of projects permits more specific insights to be gained regarding the commercial adoption, as opposed to investigation, of object technology. A total of 31% of the firms responded that they possessed an active or completed O-O project. All except one had commenced during the calendar year 1990—38% of them in the two months prior to the survey. The oldest had been in existence for four years.

Most projects, as might be anticipated, involved small teams. The average team size was reported to be six; although one very large project employed 120 people. When asked if they planned future O-O projects, thirty firms (out of the 51 who answered this question) answered in the affirmative—with planned commencement dates between twelve and eighteen months after the survey date.

Methodologies

In this section, the organisations were asked to report upon any standard methodologies they employed, whether they have examined the object-oriented approach, and how they viewed its future application to their department.

The results indicate that currently only 50% of respondents employed a standard analysis and design methodology and that these varied from traditional structured analysis and design to in-house development methodologies. Of respondents, 30% claimed their organisation had examined the object-oriented approach in some way, yet only 16% knew of any projects that had actually involved object-oriented methods. The earliest of the projects began in 1989, and, as far as having plans to employ object-oriented methods in the future, the companies were split 50–50. In general, there was a positive attitude to the future usefulness of object-oriented techniques, with 48% of respondents seeing them as having a promising or very promising future (Fig. 7), while only 20% saw them has having only some or no future. This appears a somewhat guarded response for an approach that has aroused so much attention recently, but probably reflects the lack of hard data on the promised benefits of the approach and the lack of big name products that have yet to emerge and prove their performance capabilities in the real world. The continuing recession that began in 1989/90 is probably also a major contributory factor to the apparent slow uptake since money for R&D projects will not be as forthcoming in hard economic times, as personal communications from industrial representatives testify.

In the area of object-oriented methodologies, there remains scant evidence of it being applied to anything but prototype or research systems. The approaches so far adopted appear to be ad hoc ones, using traditional tools in some sort of object-oriented manner. However, the increasing number of books and articles on object-oriented analysis and design[1,19,20,21] reflects the growing need for design tools, notations, and methodologies that can be applied to real systems development projects. It would appear that there is still some way to go in educating, researching, and applying object-oriented methodologies before they will be adopted by commercial MIS departments.

CONCLUSION

In summary, these preliminary results indicate that, while

methods were being integrated into the design process. We were therefore more interested in the type of language that was being used to implement object-oriented designs than in the name of the specific language used, and more interested in the approach taken to integrating object-oriented concepts in the design process than the specific techniques used.

The survey therefore lacked a large array of detailed questions relating to specific methodologies and techniques. This was, in part, due to the feedback received from a first draft questionnaire that was piloted on several large organisations. It was felt by many organisations that questions specifically targeted at their current methodology were likely to impinge upon what were considered by many to be commercial secrets. Detailed questions on methodologies may therefore have resulted in a greatly reduced return rate and it was felt that, while methodologies were an important area, the specific detail of the methodology was not required to attain the objectives of the study.

Several drafts of the questionnaire were pretested with large organisations. The testing consisted of unstructured interviews with several employees of the company that were familiar with the company's work in the area of object-oriented development. The questionnaire was sent to the company prior to the interview so that they had time to discuss and evaluate the questions. The interviews generally took the form of a two-part discussion. First, there was a general discussion on the area of object-oriented development, its role in the organisation, the way it was viewed by managers and staff, and their areas of interest and concern with the approach. This part of the discussion was aimed at picking up concerns and interests of management that might not have been considered in the original questionnaire. The second part of the interview consisted of running through the questionnaire and discussing specific questions: how they were worded, who would be answering the questions, which questions were more and less relevant from their point of view, which questions they would be unable to answer, and so on. The questionnaire went through several such iterations before a final version was settled upon.

The final structure of the questionnaire consisted of six sections, five of which were answerable even by those not undertaking any object-oriented development. The first section asked about course requirements, with questions on preferred length and content. This would enable educators and researchers to gain a clearer understanding of where companies were in terms of training their staff in object-oriented methods. The answers would also reveal in which areas of the lifecycle managers felt the priorities lay initially in moving toward object-oriented methodologies. The second section provided some background information on the department, organisation, or group to which the questions related while the third section was concerned with discovering the types of methodologies employed and the degree of interest companies had in the object-oriented approach. The fourth section was for those groups that had undertaken some object-oriented systems development and asked questions relating to the lifecycle approach taken (hybrid or pure), the techniques employed, the use of the methods in production systems rather than test systems, and their view of the future of object-oriented development. The fifth section asked about the information sources they used and the importance attached to the different sources identified. By understanding what types of references and published material industry practitioners use to understand the object-oriented approach, we may be able to infer something about the nature of the information being received and thus gain a perception of how the approach is being viewed. The sixth section asked for correspondence details and whether the company would be interested in a follow-up interview. We intend to follow up with more in-depth interviews with several of the companies to find out more about their specific application of object-oriented technology.

> ***"It appears that the aim of many of the larger organisations is not to develop a "standard" object-oriented methodology for all systems development but rather to train smaller groups in the "philosophy" of the object-oriented approach."***

The questionnaire was circulated to a large number of firms in the Sydney metropolitan area, with over 4,000 questionnaires being distributed in total. There was a wide variety in organisational size, and many of the larger IS operations within the area were included. The questionnaire contained sections on organisational background so that the results could be analysed by organisational type.

PRETEST INTERVIEW RESULTS

In addition to the survey, several organisations agreed to be interviewed in greater depth. Preliminary results from

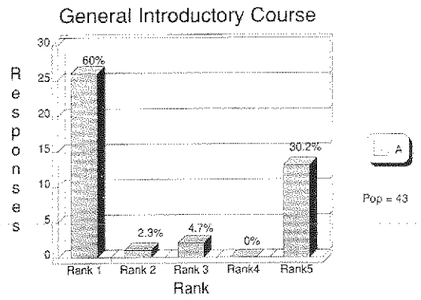


Figure 1. Percentages of respondents who placed the need for a general introductory course in each of the five ranking positions.

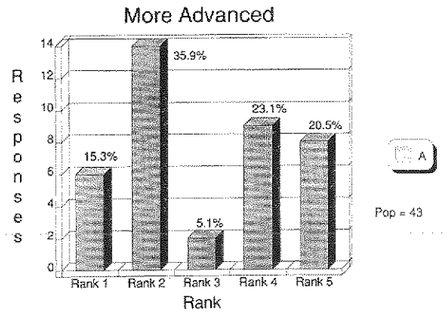


Figure 2. Percentages of respondents who placed the need for more advanced courses in each of the five ranking positions.

this small number of case studies indicate (1) a desire to acquire knowledge of object-oriented technologies, especially object-oriented applications products; (2) a realisation that reusability as a goal is now achievable with object-oriented development of corporate libraries;[17] and, frequently, (3) that such knowledge is giving the organisation a "commercial advantage" such that consequently detailed information is not permissible.

It appears that the aim of many of the larger organisations is not to develop a "standard" object-oriented methodology for all systems development but rather to train smaller groups in the "philosophy" of the object-oriented approach. These people can then be used as internal consultants and educators for the rest of the organisation. The feedback from this stage will influence how the object-oriented methodologies develop and become integrated with current systems development methodologies.

An important point that was emphasised by several organisations was that the object-oriented approach represents a quite different view of systems design, and it is this change in view, or mind set, that was of prime importance in the move to object-oriented techniques, rather than simply learning a new set of techniques. This seemed to reinforce our approach of concentrating upon the way in which the object-oriented methods were being applied within organisations rather than concentrating on discovering the techniques and tools being used.

Many companies had a generally favourable view of the object-oriented approach. However, it was noted that systems initially developed by a small trained group had to show real commercial advantages in areas such as productivity and maintenance if the techniques were to be employed on a large scale. Metrics will therefore be an important element in the argument over design methodologies, especially for larger organisations. Work is therefore required to analyse the usefulness of traditional metrics for object-oriented systems.

SURVEY RESULTS

Background

There was a wide range in the size of the organisations that responded to the survey (a total of 56 companies* in NSW), from the one- and two-people consulting companies to the 300+-people MIS shops. The majority of organisations who responded had 0–20 people working in the area of information systems. The companies were asked to specify the percentage of systems designed and purchased to discover how important design methodologies were to their software development effort. Without an understanding of the importance of design methodologies to companies, we cannot judge with any accuracy how the object-oriented paradigm is viewed in industry. The results indicate that on average 66% of systems were designed rather than purchased (total sample size 53). This indicates that design methodologies do play an important part in the software development process in the majority of companies sampled, thus representing companies to which object- oriented methodologies may be relevant.

It has been suggested that the object-oriented approach may be particularly useful for "programming in the large." Therefore, a question on the size of the systems designed was included. This would allow the size of systems developed to be correlated with the methodology employed. Thus, for those systems that were designed in-house, the percentage that were large (greater than 100,000 source lines of code [SLOC]), medium (between 5,000 and 100,000 SLOC), and small (less than 5,000 SLOC) were recorded. The results indicated a relatively even spread in the size of systems developed—20% being classified as large, 42% as medium, and 38% as small.

In summary, the sample encompassed a wide variety of organisation sizes and types that designed many sizes and types of systems. The organisations sampled seemed to be those to which the object-oriented paradigm was applicable, namely those that designed many of their own systems.

* A problem with the questionnaire that may have affected the response rate was its title: "Object-Oriented Questionnaire." This may have put off some potential respondents immediately, so it is hard to draw any conclusions from the relatively low response rate.

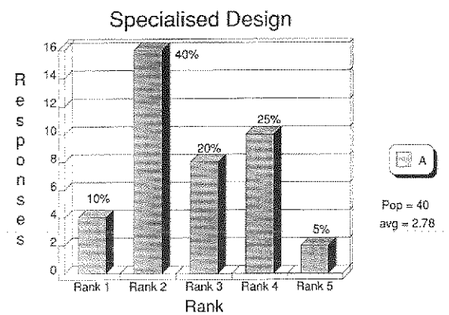


Figure 3. Percentages of respondents who placed the need for more specialised design courses in each of the five ranking positions.

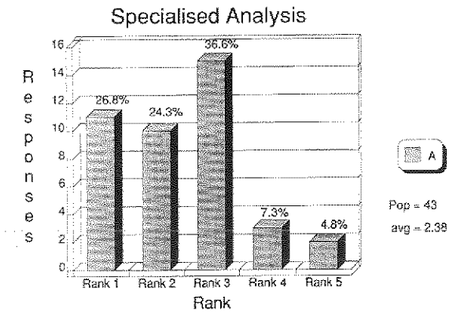


Figure 4. Percentages of respondents who placed the need for more specialised analysis courses in each of the five ranking positions.

Courses

One of the major ways that companies learn about new technologies, methodologies, and research areas is through professional short courses. By allowing companies to specify their needs and demands for object-oriented courses in the survey, we were able to infer the current state of industry knowledge on the topic and identify those areas that appear of particular concern.

In response to a simple yes/no question as to whether the respondents would be interested in receiving information on object-oriented courses, 90.5% indicated they would, while only 7.5% indicated they would not. This indicates a significant amount of interest in the paradigm generally and in learning more of what it has to offer. The most popular desired course length was between three and five days, again indicating a significant interest to invest time and money in developing a greater understanding of the paradigm. It was also found that 53.5% of the companies had already sent people to various object-oriented courses covering the topics of analysis, design, and programming.

We then asked what type and level of course was desired and in what order of priority the courses were viewed. The results indicated a clear need for general introductory courses (Fig. 1), with over 60% placing a general introductory course as their top priority. Interestingly, the next highest percentage of responses (30%) put a general introductory course as their lowest priority. This reflects the strength of the two ends of the spectrum as far as the object-oriented approach is concerned: those who have been involved for some time and are looking to refine particular skills for use in production systems and those who are just beginning to learn about object-oriented systems development. Similarly, yet a little more ambiguously, a more advanced course was viewed either as a relatively high priority or as a low priority (Fig. 2), depending on which end of the spectrum the firm was from. An interesting result emerged concerning the demand for more specific courses, such as ones aimed only at programming: 50% of respondents put specialised design courses (Fig. 3) in their top two priorities, 51% put specialised analysis courses (Fig. 4) in their top two priorities, while only 15% put specialised programming language courses (Fig. 5) in their top two priorities. This seems to indicate a greater concern with the earlier stages in the lifecycle than may have been anticipated and may provide some evidence that the merging of object-oriented methodologies with more traditional ones will be of the object-oriented analysis/object-oriented design followed by functional implementation variety as discussed by Henderson-Sellers.[16] Indeed, despite, or possibly because of, the greater emphasis in recent years on object-oriented languages, 58.5% of the respondents placed specialised language courses in the bottom two priorities for courses. This is an interesting feature of industry's demand for courses when one considers the amount of literature that has been devoted to language issues. Overall, 52% had a general introductory course as their top priority, with programming the least popular at 6% (Fig. 6). It has been suggested that this is because large COBOL shops are unlikely to take on OOP until IBM or another large vendors "legitimizes" the OOP market. With the more recent formation of the Object Management Group,[18] which now includes within its membership all such large vendor companies, such concerns should by now have evaporated. Similarly, the concern with analysis and design could be a reflection of the recognition that the major costs of systems are in these stages rather than in the coding phase.

The results are by no means conclusive in identifying how industry views the object-oriented paradigm, but they do provide some hard evidence that object-

≡ Your Turn ≡

How about it? How are we doing? This is the place to send us your message. Tell us what you like and what you don't, what you'd like to see but don't — or whatever other feedback you'd like to send. Send comments to Robert Shelton, 1850 Union Street, Suite 1584, San Francisco, CA 94123; Fax: (415)928-3036; or, if you're ordering, send along with your order.

Please check whichever box applies:

	Read entirely	Scanned	Found helpful	Will refer to	Didn't read
Feature	❑	❑	❑	❑	❑
Retrospective	❑	❑	❑	❑	❑
Technology Transfer	❑	❑	❑	❑	❑
Interview	❑	❑	❑	❑	❑
FYI	❑	❑	❑	❑	❑
Product Announcements	❑	❑	❑	❑	❑

Please rate the following ***1*** (dislike or strongly disagree) to ***10*** (like or strongly agree):

Presentation of material __ Comments: _____

Accuracy of material __ Comments: _____

Overall helpfulness in your job __ Comments: _____

What else would you like to see in THE HOTLINE? _____

Any other comments? _____

_____

_____

Name, title & phone (optional): _____

hotline on
OBJECT-ORIENTED
technology

❑ **Yes,** plug me into the latest thinking and developments in object-oriented technology. Enter me as a subscriber at the term marked below and rush me the current issue. This is a risk-free offer — I may cancel my subscription at any time and promptly receive a refund for the unused portion.

1 year (12 issues) ❑ $249 *(outside US add $30 per year for air service)*

2 years (24 issues) ❑ $478 *(save $20)*

Back issues @ $25 each *($27.50 foreign)*: Vol.1, Nos. _____ Vol.2, Nos. _____ Vol.3, Nos. _____

❑ **Phone order**
Call Subscriber Services
(212)274-0640

❑ **Fax order**
Fill out term and name/address information, then fax to:
(212)274-0646

❑ **Send me a copy of**
THE INTERNATIONAL OOP DIRECTORY
@ $69 ($81 foreign)
(all directory orders must be prepaid; fax or mail credit card information to SIGS, 588 Broadway, #604, NYC 10012; make check payable to OOP Directory. Foreign orders must be prepaid in US dollars drawn on US bank.)

❑ **Bill me**
❑ **Check enclosed**
❑ **Credit card orders**
❑ MasterCard
❑ Visa

Mail (and make out check, if applicable) to:
THE HOTLINE
Subscriber Services
P.O. Box 3000, Dept. HOT
Denville, NJ 07834
(foreign orders must be prepaid in US dollars drawn on a US bank)

Card# Expiration Date

Signature

Name

Company

Street/Building#

City/Province

ST/Zip/Country

Telephone

Send me a complimentary copy of your related publication(s):

❑ THE JOURNAL OF OBJECT-ORIENTED PROGRAMMING

❑ OBJECT MAGAZINE

❑ THE C++ REPORT

Call Subscriber Services
(212)274-0640
or send order to
SIGS, 588 Broadway, #604
New York, NY 10012

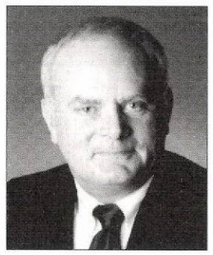
D2BC

hotline on
OBJECT-ORIENTED
technology

VOLUME 3, NUMBER 4 THE MANAGER'S SOURCE FOR TRENDS, ISSUES & STRATEGIES FEBRUARY 1992

The big prize: acceptance of O-O by the MIS community

Readers of Hotline will be more aware than most that the object-oriented marketplace is coming along nicely.



Donal O'Shea

Development of the market took a noticeable step forward in the second half of 1991 as the number of mentions of object technology in the business press surged. The most notable was the "Software Made Simple" cover story devoted to O-O in *Business Week* last September.

The O-O market today is similar to the UNIX market as it was in 1984. When AT&T decided to start pushing UNIX, they spent tens of millions of dollars advertising UNIX System V, not in the trade press, but in the business press: *Business Week, The Wall Street Journal, Forbes*. By causing corporate executives to demand of their MIS directors, "What is this System V stuff?" this advertising strategy did more than anything else to push the commercial computing community towards UNIX. It became acceptable to spend to explore UNIX.

In mid-June 1991, references to "object-oriented" in business press reports of the joint venture arrangements between IBM and Apple began to have the same effect. The commercial world is becoming sensitized to the existence and potential benefits of object-oriented technology. The snowball has started to roll down the hill.

From a vendor's viewpoint, O-O will only have arrived when the MIS community accepts it for everyday use. The scientific, engineering, and technical market is big, but it is not homogeneous. The fragments are small and require lots of customization. The big opportunity for the independent software company is in the tens to hundreds of thousands of copies of a program that the MIS market will purchase.

Despite a good beginning, it could take a long time before the MIS community completely accepts O-O. Some would argue that to this day UNIX has not really been accepted by the MIS community! Let's take a look at what's going on with O-O in MIS.

THE MIS COMMUNITY MUST ADOPT O-O FOR ITS OWN SAKE

While the benefits of UNIX for MIS may be a somewhat oblique freedom from the tyranny of vendor dependence,

continued on page 4 ...

IN THIS ISSUE

FEATURE — The big prize: acceptance of O-O by the MIS community *Donal O'Shea*	1	2	FROM THE EDITOR
RETROSPECTIVE— 1991—The year it all changed *Tom Love*	7	21	PRODUCT ANNOUNCEMENTS
TECHNOLOGY TRANSFER— Making the transition to O-O technology *Ron Suarez, PhD*	9	22	NEW PARTNERINGS & ACQUISITIONS
INTERVIEW— From classroom to shipped product in 20 months *Ron Suarez, PhD*	13	23	CALENDAR
FYI — Industry publication excerpts	18	24	YOUR TURN

From the Editor



As this is the first issue of *Hotline* that I have edited, a few words on direction and future topics would be in order. We will bring you late-breaking news and important industry trends, stories about real-world experiences with object technology, frank insights from seasoned colleagues, and discussions of issues that managers should know about to successfully implement object technology. I'll elaborate in a moment but, before doing so, let's take a look at this month's issue.

In the feature, Mr. Donal O'Shea of Servio Corporation addresses the big win that providers of object technology and object-based products covet: acceptance by corporate MIS. On the whole, commercial software product builders have been easier to convince than MIS departments. They see different economic signals than MIS, and have to take risks or be second to market with their products—and in today's economy, second is losing. With many MIS departments smarting from the limited returns they're able to show from huge investments in CASE and mass training in structured methods, their "I don't want to miss out, but I'll wait and see" attitude is understandable. As Mr. O'Shea's examples suggest, however, "wait and see" is giving way to exploration and commitment.

Users developing network-based applications for themselves because MIS developers "take too long," "are too restrictive," and "don't understand our business" are intruding on the once insular world of MIS. This trend reflects a deadly combination of real needs, expectations, and seductive technology. William F. Perrin of VARICAST calls these systems "islands of information" because they're separated from the rest of the business. Such systems don't do what a skilled MIS team can: make information sharing possible. Redundant, unvalidated, unmaintained—and often not backed up regularly—these island systems hold a frightening amount of mission-critical data! One large U.S. electric utility with which I am familiar runs the entire budget and financial management database for their nuclear power operation on a PC dbase server—undocumented and overseen by two programmers with no experience in production applications, version control, formal methods, database design, or electric-utility finance. Implementation (to the point that the users stopped screaming about the software) took two years. Two user-driven replacement attempts have also failed—for many of the same reasons that MIS projects often land on the rocks! So much for the users themselves doing it faster and better! Corporate MIS is now taking over the project. Will corporate try a fresh approach, or will they fall back on the mainframe and DB2? The jury's not in yet. Read Mr. O'Shea's story for some insights on what other large corporations are doing. His article addresses many of the issues with which this utility—and probably your company as well—still struggles!

Mr. Tom Love brings us an industry retrospective for the second year running. As his article (and Mr. O'Shea's as well) points out, one of the most significant changes during 1991 was seeing object oriented technology in the nontrade press. When a new technology becomes a common business term, technologists face a demand upswing—and a fresh opportunity to fail as we have done before if we do not take our own learnings under advisement. We information professionals can be our own worst enemy! Our credibility as professionals is once again on the line—our effectiveness with object technology concepts will determine our success...or failure—in the eyes of the users and buyers of our wares for the remainder of this decade. This time, however, the advent of visual programming languages and "user programmable" reporting and database tools has changed the balance of power. Although our organizations may not benefit from island systems, our customers now have a choice and "do it yourself" is increasingly an attractive option. As our industry matures, it will remain our personal and professional challenge to continuously prove to our customers that we add value to their business.

Ms. Beatriz Infante of Momenta Corporation speaks to us about a very successful application of object technology to a shrink-wrapped commercial prod-

Editor
Robert Shelton

SIGS Publications, Inc. Advisory Board
Tom Atwood, *Object Design*
Grady Booch, *Rational*
George Bosworth, *Digitalk*
Brad Cox, *Information Age Consulting*
Chuck Duff, *The Whitewater Group*
Adele Goldberg, *ParcPlace Systems*
R. Jordan Kreindler, *General Electric*
Meilir Page-Jones, *Wayland Systems*
Tom Love, *Consultant*
Bertrand Meyer, *Interactive Software Engineering*
Sesha Pratap, *CenterLine Software*
P. Michael Seashols, *Versant Object Technology*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

Hotline Editorial Board
Jim Anderson, *Digitalk, Inc.*
Larry Constantine, *Consultant*
Mary E.S. Loomis, *Versant Object Technology*
Reed Phillips, *Knowledge Systems Corp.*
Trygve Reenskaug, *Taskon A/S*
Zack Urlocker, *Borland International*
Steven Weiss, *Wayland Systems*

SIGS Publications, Inc.
Richard P. Friedman, Founder & Group Publisher

Art/Production
Kristina Joukhadar, Managing Editor
Susan Culligan, Creative Director
Elizabeth A. Upp, Production Editor
Caren Polner, Desktop Designer

Circulation
Diane Badway, Circulation Business Manager
Kathleen Canning, Fulfillment Manager
John Schreiber, Circulation Assistant

Administration
David Chatterpaul, Accounting
Frank Ferrantelli, Accounting
Suzanne W. Dinnerstein, Conference Manager
Laura Taylor, Conference Coordinator
Jennifer Fischer, Assistant to the Publisher
Jennifer Englander, Administrative Assistant

Margherita R. Monck, General Manager

Jane M. Grau, Contributing Editor

hotline on OBJECT-ORIENTED *technology*

The Hotline on Object-Oriented Technology (ISSN #1044-4319) is published monthly by SIGS Publications, Inc., 588 Broadway, NY, NY 10012, (212)274-0640.

© Copyright 1992 SIGS Publications, Inc. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the U.S. Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher.

Mailed First Class. Subscription rate — one year (12 issues) $249, Foreign and Canada $279. Single copy $25.

POSTMASTER: Send address changes & subscription orders to The Hotline, Subscriber Services, P.O. Box 3000, Dept HOT, Denville, NJ 07834.

Submit editorial correspondence to Robert Shelton, 1850 Union Street, Suite 1548, San Francisco, CA 94123. Voice: (415) 928-5842; fax: (415) 928-3036.

SIGS PUBLICATIONS

Publishers of *Hotline on Object-Oriented Technology, Journal of Object-Oriented Programming, Object Magazine, The X Journal, The C++ Report, The Smalltalk Report,* and *The International OOP Directory.*

Back issues available

Vol.3, No. 3/January '92 ≡ Enterprise object modeling: knowing what we know • *Shelton* ≡ Adopting objects: pitfalls • *Connell* ≡ Adoption rate of object technology: a survey of NSW industry • *Henderson-Sellers*

Vol.3, No. 2/December '91 ≡ Accepting object technology • *Bennett* ≡ Adopting objects: a path • *Connelli* ≡ Incorporating graphical content into multimedia presentations • *Berman & Weiss*

Vol.3, No. 1/November '91 ≡ Leading the U.S. semiconductor manufacturing industry toward an object-oriented technology standard • *Hollowell* ≡ Coping with complexity: OOPS and the economists' critique of central planning • *Lavoie, Baetjer, & Tulloh* ≡ Choosing Object Technology: What's the object? • Rosenthal ≡ OOP: the MISsing link • *Rovira*

Vol.2, No. 12/October '91 ≡ A modest survey of OOD approaches • *Bulman* ≡ What is a "certified" object programmer? • *Bellin* ≡ Perspective: investing in objects today •Bowles ≡ Object oriented in Melbourne, Australia • *Haebich* ≡ The Object Management Group • *Gutman & Matthews*

Vol.2, No. 11/September '91 ≡ From applications to frameworks • *Urlocker* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Getting started with object technology: efffectively planning for change • *Lorenz* ≡ Object statistics on the way • *Lenzi* ≡ On objects and bullets • *Onart*

Vol.2, No. 10/August '91 ≡ Distributed object management: improving worker productivity• *Osher* ≡ Getting the best from objects: the experience of HP • *Coleman & Hayes* ≡ Applications: EC employs object technology ... • *Stein* ≡ Capacity Planning: Fiddling while ROMs burn • *Rovira*

Vol.2, No. 9/July '91 ≡ Multimedia is everywhere! • *Weiss & Berman* ≡ Developing an object technology prototype • *Kulinek* ≡ Object-oriented capacity planning • *Rovira* ≡ How OOP has changed our developmental lifecycle • *Zeik* ≡ Modularization of the computer system • *Angenius*

Vol.2, No. 8/June '91 ≡ Domain of objects: the Object Request Broker • *Dyson* ≡ Object-based approach to user documentation • *Reeder* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Do we need object-oriented design metrics? • *Hopkins*

Vol.2, No.7/May '91 ≡ Hybrid object-oriented/functional decomposition for software engineering • *Henderson-Sellers* ≡ So, what makes object databases different? (Part 4) • *Blakey* ≡ Using the generic application to solve similar domain problems • *Gossain* ≡ Experiences using CLOS • *Hopkins* ≡ International Conference on Object-Oriented Technology, Singapore • *Bennett*

Vol.2, No.6/Apr. '91 ≡ An artist's perspective of programming • *Rovira* ≡ So, what makes object databases different? (Part 3) • *Blakey* ≡ Moving from Pascal to C++, Part 3 • *Gole* ≡ Object projects: what can go wrong • *Stewart* ≡ Reflections from LOOK-'91 • *Anders-Angenius*

Vol.2, No.5/Mar. '91 ≡ Large-scale object systems development • *Morgan & Davis* ≡ Data definition languages • *Blakey* ≡ Object disorientation • *Kerr* ≡ Moving from Pascal to C++, Part 2 • *Keller* ≡ The Object Management Group • *Barber*

Vol.2, No.4/Feb. '91 ≡ Moving from Pascal to C++ • *Sollisch* ≡ The Australian object-oriented scene • *Henderson Sellers* ≡ So, what makes object databases different? • *Blakey* ≡ Managing the evolution • *Nowobilski* ≡ 1990: the year of realism • *Love* ≡ Choosing a configuration for object development • *Moody*

Vol.2, No.3/Jan. '91 Special issue: CASE and Methodology ≡ Object orientation: stop, look and listen! • *Page-Jones* ≡ The spiral model for object software development • *Wasserman & Pircher* ≡ Bootstrapping object-oriented CASE • *Wybolt* ≡ A CASE tool for object software development • *Ackroyd* ≡ Comments on "Race for the database pennant" • *Loomis*

Vol.2, No.2/Dec. '90 ≡ Waiting for OODB2 • *Durham* ≡ Object modeling • *Singh* ≡ Object-oriented COBOL — the next generation • *Adams & Lenkov* ≡ Object-oriented training in the UK • *Hopkins*

Vol.2, No.1/Nov. '90 ≡ Objects by reamwork • *Constantine* ≡ Race for the database pennant: a third-generation manifesto • *Stein* ≡ Project team descriptions *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.12/Oct. '90 ≡ Crown jewels: the value of OOT in MIS departments • *Bonar* ≡ Why is an object database faster than a relational system? • *Loomis* ≡ Visual programming object language environments • *Klausner* ≡ Hype — necessarily (or a necessary) evil? • *Adams* ≡ On Managing: Déjà Vu • *Lenzi*

Vol.1, No.11/Sep. '90 ≡ Introducing OOPS through design — it's all in the words • *Copen Goldstein* ≡ The Australian object-oriented scene • *Henderson-Sellers* ≡ Objects on tour (Review of Borland's OOP World Tour) • *Lenzi* ≡ Design principles for modular product development • *Bennett* ≡ Asset management and object-oriented technology • *Hopkins & Warboys*

Vol.1, No.10/Aug. '90 ≡ WyCASH+: An application built within an OOP environment • *McCullough & Deshler* ≡ Mathematics and methodology • *Martin* ≡ Organizing for reuse • *Goldberg & Rubin* ≡ CALS Expo–89 Conference • *Purdy*

Vol.1, No.9/July '90 ≡ Cognos and Eiffel: a cautionary tale • *Leathers* ≡ What's in an object anyway? • *Novobilski* ≡ The impact of Windows 3.0 on the OOP market • *Urlocker* ≡ Introducing reuse into the development process • *Goldberg & Rubin*

To order, see back page; for reprints of individual articles, call Michael Biggerstaff at Reprint Management Services (717) 560-2001.

Calendar

The Hotline Calendar presents conferences and meetings that focus exclusively on object-oriented technology. To have a meeting or conference listed, please send the dates, conference name and location, sponsor(s) and contact name and telephone number to the Editor: Robert Shelton, 1850 Union Street, Suite 1584, San Francisco, CA 94123; Fax: (415) 928-3036.

February 3–5, 1992	February 5–7, 1992	February 11–21, 1992	February 25–28, 1992	March 23–26, 1992
LOOK '92	**OOP '92**	**C++ Across America**	**Software Development '92**	**DB/Expo '92**
Copenhagen, Denmark	Munich, Germany	SF, LA, Dallas, Chic., Bos., NYC, Wash. D.C.	Santa Clara, CA	San Francisco, CA
Sponsor: Niveau, *Object Magazine*, & *The Journal of Object-Oriented Programming*	Sponsor: *The Journal of Object-Oriented Programming & Object Magazine*	Presented by: *The C++ Report*	Sponsor: Miller Freeman	Sponsor: NDN
Contact: Miles Silverman (212) 274-0640	Contact: Miles Silverman (212) 274-0640	Sponsored by: Microsoft	Contact: (415)905-2414	Contact: (415)941-8440
		Contact: (212)274-9135		

C++ Language System Release 3

UNIX System Laboratories, Inc.
190 River Rd.
Summit, NJ 07901
(908) 522-6000
(800) 828-UNIX

UNIX System Laboratories announced its **C++ Language System Release 3,** the cornerstone of USL's C++ product family. Developed jointly by UNIX System Laboratories and AT&T Bell Laboratories, applications written using the previous releases (2.0 and 2.1) of the USL C++ Language System are upward compatible with Release 3. The templates feature, also known as parameterized types, utilizes Bjarne Stroustrup's design, as included in the draft ANCI C++ standard. It incorporates and extends the initial implementation of class templates by Object Design, Inc. The source code for the C++ Language System, Release 3 has been developed and tested on multiple hardware platforms running versions of UNIX System V and SunOS. This product is designed to be easily ported to UNIX platforms and has been ported to other operating systems as well. A new document, the C++ Language System Release 3 Internals Manual, provides information to assist USL's C++ licensees in extending or modifying the C++ Language System, and discusses the design of major components of the language system, including the templates feature.

New Partnerings and Aquisitions, February, 1992

Borland named **Kelsey Selander** as Borland's new vice president of marketing. Selander was formerly a vice president at Lotus Development Corp.

NTT DATA COMMUNICATIONS SYSTEMS, the Japanese systems integrator, made cooperative technical and marketing contracts with **UniSQL, Inc.** NTT DATA is cooperating with UniSQL to address its future database management system needs and the needs of its system integration customers, especially in preparation for the shift to multimedia applications and downsize systems.

AICorp, Inc. today announced that it purchased its affiliate **AICorp Europe** from Schroder Ventures, the international venture capital firm. **Schroder Ventures** will receive 800,000 shares of AICorp common stock in return for selling its stake in AICorp Europe to AICorp, Inc. Subject to the terms of the agreement, Schroder has agreed not to sell 84%, or 672,000, of the 800,000 shares for three years; the remaining 128,000 shares are restricted to one year.

Novell took an equity position in **Serius Corporation** totaling $700,000. A Novell executive, yet to be named, will be appointed to Serius' Board of Directors. Further details of the agreement have not been made public at this time.

The European Economic Community's **ESPRIT II** "Business Class" project has selected the Eiffel environment as the primary software platform. The project's goal is the production of object-oriented CASE tools that address analysis, design, implementation, and maintenance needs for business applications.

Two-year-old start-up **Cayenne Systems** was recently re-christened **Gain Technology**. Gain and its silent partner **Matsushita Electric**, claim a two-year technology lead over other multimedia projects. Both companies have been working secretly for two years on the technology that has given Cayenne its new name: the **Graphical Authoring Initiative**. Apparently, Matsushita has no equity position in Gain, however, their exact financial arrangement have yet to be detailed.

HyperDesk announced its Premier Partners Program to meet the needs of early customers. The program includes development consulting and early access to initial product and future upgrades.

Interactive Software Engineering signed a distribution agreement with **SIG Computer**, Braunfels, Germany, the developer of Eiffel/S. Written in Eiffel, Eiffel/S is a compiler for the object-oriented language and translates to C source. The current version implements Eiffel 3. Interactive Software Engineering is the distributor for the United States, Canada, Australia, and the Pacific Rim.

KPMG . ExIS and the **MacApp Developers Association** (MADA) announced that IcePick, ExIS' newly developed view editor and development environment, will be exclusively published and distributed by the MacApp Developers Association. By publishing and distributing the product, MADA will be able to provide a discount to members. KPMG . ExIS is a business unit of **KPMG Peat Marwick**, formed as a result of a joint business alliance between KPMG Peat Marwick and **Apple Computer**, Inc. MADA is an independent nonprofit professional association of MacApp developers.

TauMetric Corporation acquired exclusive rights to **Oregon Software's** line of products. Oregon Software's sales and support services will be located in TauMetric's San Diego office by January 1992. TauMetric wrote the C++ front end for the Oregon Software C++ compiler and, since 1990, has been responsible for development and technical support of Oregon Software products. Established in 1983, TauMetric initially was a consulting firm specializing in computer software. The company later expanded into the development of a number of proprietary products.

Rational (Santa Clara, CA) set up four offices in European countries and is offering various consulting services and special software assemblages in addition to its traditional development systems. Rational has announced a cooperative marketing agreement with O2 Technology under which the companies will jointly market their products to object-oriented software development organizations. Under the cooperative marketing agreement, the two companies will explore opportunities to produce a complete integration of Rational Rose, rational's graphical object-oriented analysis and design tool, and O2's OODBMS. Rational has also teamed up with Tartan, Inc. (Monroeville, PA) to provide a joint offering, linking Rational's development system with Tartan's Ada cross-compilers.

uct. For those doubting the viability of object technology or pure-object environments for real-time applications, Momenta's story should be a word to the wise. With the cost-per-million compute cycles dropping by over 50% per year, and the effective power available in a notebook package growing explosively, our industry sorely needs to shake off the vestiges of the "programmer as the code optimizer" mentality. Application performance is not unimportant (you should hear me protest when my computer is laggardly!), but Ms. Infante points to sound design and clarity of purpose as the primary determiners of overall product performance. Furthermore, today's definition of performance has much more to do with meeting time-to-market deadlines and customer requirements than the optimal use of every computer cycle. Selecting the right delivery technology and approach are more important to a well-behaved, well-received product than is the traditional vision of optimization taught to computer science students.

Dr. Ron Suarez of Arbor Intelligent Systems, Inc. picks up this theme in his consideration of how to transition successfully to object technology. Learning the frame of mind that underlies object-oriented technology is more significant than the language itself. The challenges are human, not technological. The greatest hurdle in the transition from traditional to object technology is changing ourselves. Ultimately, success or failure in the way the change is undertaken can determine the company's showing in the marketplace.

As weavers know, a piece of woven cloth is composed of longitudinal strength members called the warp and opposing connective members called the woof. Warp runs consistently through the length of the piece, while the woof may vary to give texture, meet a particular need or accommodate the selection of yarn available at the moment. I found a similar pattern in this month's articles. Among the common threads binding our authors perspectives:

- Object technology may have a greater impact on market position and time to market than any other computer technology.
- Real commercial products are being built using object technology—the technology is "here" for those making the learning investment.
- Object-oriented technology is not a language.
- Adopting object technology is a "people change" problem, so buying a C++ compiler and learning the language (instead of the concepts) can be a step backward.
- Discipline in the development process is more critical with object technology than it was with procedural technology, yet many proponents of object products would have you believe that prototyping is the discipline of choice.
- When selecting object products, determine in advance what you're looking for. If you're shopping for the first time, ask people who have bought before. Use others' experience. Remember that responsible vendors and practitioners will help you understand the limits, risks, and problems as well as the benefits.

Yet like woven cloth, you'll see different solutions, approaches, and viewpoints. Our authors agree that discipline is required, but a consensus on the structure of that discipline is no more present here than in the industry as a whole. I believe such a structure is emerging, but many practitioners who offer methods and tools still don't understand the full scope of the problem that users of object technology need to solve. Revamped structured analysis, as several of our authors point out, seems less appropriate than some fresh, new ways of looking at our customers' needs and our work. Also, while our authors address common themes of profitability and quality it becomes clear that different organizations will adopt object technology for different, business-driven reasons. Arguments which favor using objects in other organizations may have more or less impact in yours.

Our effectiveness with object technology—as in so much in life—is a function of our span and our depth. Those working with matters technical find it easy to specialize. Western cultures emphasize depth at the expense of breadth. Over the next year, I intend to expand our vision. We will tap the experiences of practitioners, thinkers, and vendors. We will look beyond the technology itself, exploring the issues that form the environment in which the technology operates. This I find most exciting, as it gives me—and I hope, you as well—opportunities to explore the as-yet undiscovered. In the upcoming year, the topic areas our authors will be addressing include:

- objects as enablers to client-server, distributed architecture
- welding object applications to today's relational databases
- evaluating and using object database management systems
- O-O implementation strategies and tactics
- business, social, and legal issues surrounding O-O
- Microsoft vs. Borland—who's really winning here?
- de facto and emerging standards for O-O
- modeling, methods, and tools
- selecting object products, tools, training, and consulting services
- managing risk and facing failure
- successful project management
- objects and visual programming
- multimedia applications made more achievable by objects
- the growth of hybrid languages for MIS, such as O-O COBOL

This is a start! Are there issues perplexing your organization? Do you have experiences that your peers should be aware of? As editor, my responsibility extends well beyond making opinionated remarks on the inside cover and selecting themes for each issue. It would be my expectation that *Hotline* will provide a platform for communicating useful, practical experience, the successes and failures you're willing to share, and the information that consumers and providers of object technology need to make sound decisions for their respective organizations. I welcome your input, and look forward to delivering valuable industry news to your desktop in the upcoming months.



continued from page 1 ...

O-O could have a much more direct effect on the MIS directors bottom line.

O-O can bring benefits to three important areas of MIS: programmer productivity, software maintenance, and systems integration.

Programmer productivity

From the MIS perspective, the most important claim for object-oriented technology is the productivity improvement it promises in the applications development sphere. For years, we have heard of the applications backlog. This is the period it will take to catch up on the development of the applications a business needs. It always seems to be two years, and the advent of structured programming, chief programmer teams, CASE, and all the other silver bullets has done nothing to help narrow it. Object-oriented technology promises to reduce that gap.

A joint study by General Motors/EDS and Servio Corporation has shown that improvements of fourteen-to-one are achievable in terms of the programmer man-months required to build an application, in this case a factory equipment maintenance scheduling program. The elapsed time for the project was reduced by a factor of five. Using Smalltalk and GemStone, instead of PL/I and an RDBMS, the number of lines of code were reduced from 250,000 to 22,000. Other studies have shown similar productivity gains.

Software maintenance

The software maintenance cost on a central office telephone switch is in the region of a million dollars a day. No wonder the first industry to buy into object-oriented programming in a big way was the switching systems business. Using traditional programming techniques to change the North American numbering plan from a three digit area code and a seven digit central office and subscriber number (510) 555-1212 to, say, a four digit area code, would involve identifying every routine in the system that handled the ten digit number. Some cross-referencing problem!

In an O-O system, the definition of the numbering plan would be in one object. Inheritance and messaging would take care of the problem from there.

The MIS community spends a far greater percentage of resource on maintenance than they do on new application development. Consequently, the development of future applications using an O-O approach will allow the diversion of resource to new applications or an overall reduction in cost.

Systems integration

There is no doubt that object technology can offer some help in solving the problem of tying together the array of different applications running on different platforms developed by different software developers and using different development philosophies that the MIS community have to deal with. Wrappers that allow an alien block of code or a complete application to be repackaged so that it externally looks like an object, are being developed to do this.

The extent to which O-O can help in the maintenance of large, existing applications systems written in traditional languages is another question. Many MIS shops that are moving to a client-server environment and are forced to retain, say, a major COBOL application, can best retrofit the legacy application with a desktop GUI front-end using an object-oriented bridge as the glue.

There is a good argument that says only O-O will permit the implementation of the next generation of distributed computing systems. To optimize network load, it will be essential to move the data and the programming to the most appropriate computing engine for execution.

THE MIS COMMUNITY SHOULD START WITH O-O RIGHT NOW

Object technology is ready for the MIS community now. I don't advocate that everybody go out and implement a large, mission-critical application tomorrow morning. However, there are people who have already done it and are seeing the payoff.

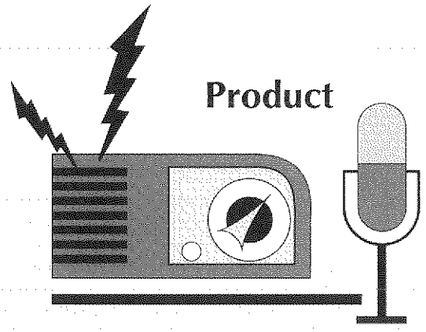
In June of 1990, the Brooklyn Union Gas Company, not exactly in the space rocket business, replaced their 15+ year-old customer information system with a new system built using O-O technology. This system is key to the collection of a billion dollars in annual revenue; by any standards, mission critical.

General Motors has installed an interactive inventory management system at a car body plant in Flint, Michigan, another critical application in everyday use. The plant runs fine, and the end users were particularly happy with the iterative development that O-O allowed; the application does what they want it to do, not just what they managed to write down in the original specification.

OBJECT TECHNOLOGY ISN'T A LANGUAGE

The Brooklyn Union Gas Company project, mentioned above, was implemented in PL/I on an IBM 3090-300E. To exploit O-O, it isn't imperative that an application be written in an object-oriented language, though it may be easier for the programmers to stick to the object paradigm if they are writing in Smalltalk than if they are using PL/I.

But, Brooklyn Gas is not the only company to



Product Announcements

Product Announcements is a service to the readers of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY; it is neither a recommendation nor an endorsement of any product discussed.

Serius Developer 3.0

Serius Corp.
1981 E. 4800 South
Salt Lake City, UT 84117
(801) 272-7788
(800) 876-6847

Serius Corporation released **Serius Developer 3.0,** an object-oriented software development environment for the Apple Macintosh. Key features include ObjecTalk: a script-based programming language that does not require a specific syntax ; ObjecSketch: an icon-based programming language in which users graphically connect icons with lines representing sequences of operations and flow of data; and Subjects: a user-definable group of objects and functions, or "super object," reusable in different applications. Serius Developer 3.0 is sold with a core set of objects and functions supporting all aspects of the Macintosh application interface, e.g., menus, buttons and windows. Serius Developer also comes with Serius Database, a bundle of objects for building relational multi-user databases. Additional object bundles designed to work with Developer 3.0 are available separately and include Serius Multimedia, Serius Communications, Serius XCMD, and Serius InterApp. In total, the Serius Object Library consists of 52 objects and 400 programming functions, ranging from comparatively simple items such as the Menu Object, to complete, compiled tools such as the database, animation, and movie engines..

HyperDesk Distributed Object Management System

HyperDesk Corporation
2000 West Park Dr.
Westborough, MA 01581
(508) 366-5050

HyperDesk Corporation introduced the **HyperDesk Distributed Object Management System** (DOMS). The HyperDesk DOMS is a suite of advanced software tools and services designed to enable application developers to build and integrate truly open distributed applications for diverse computing environments that include PC's, workstations, and servers. HyperDesk DOMS is the first product based on the Object Management Group's Object Request Broker (ORB) standard, the central communication and integration manager for software objects. The HyperDesk Distributed Object Management System Development Kit will be available in February. A runtime version of the product is expected to be released in June. The HyperDesk DOMS tools include a graphical user interface for development and testing, a dynamic interface to create and modify applications at runtime, and a definition language to create objects. HyperDesk services include an object database with repositories and class libraries for object storage, a Kerberos-based authentication service for use validation, and a location service to track objects.

Turbo C++ Borland C++ 3.0 Borland C++ & Application Frameworks 3.0

Borland International, Inc.
1800 Green Hills Rd.
P.O. Box 660001
Scotts Valley, CA 95067-0001
(408) 439-4825

Borland International, Inc. announced the availability of three new software development products that feature C++ in addition to the C Programming language. **Turbo C++** for Windows 3.0 offers an inexpensive route to Windows programming for entry-level programmers and includes a Windows-hosted development environment, ObjectWindows application framework, EasyWin, and Resource Workshop. **Borland C++ 3.0,** a third generation professional C and C++ compiler, now supports global optimization and contains all of the software included in Turbo C++ for Windows 3.0, with the exception of ObjectWindows. **Borland C++ & Application Frameworks 3.0,** as the high-end product of Borland's C++ product line, includes everything in Borland C++ 3.0, plus ObjectWindows and Turbo Vision application frameworks, complete with source code for the frameworks and the runtime library.

Energize Programming System

Lucid, Inc.
707 Laurel St.
Menlo Park, CA 94025
(415) 329-8400

Lucid, Inc. introduced its **Energize Programming System** designed to provide a natural path to object-oriented programming for UNIX programmers. The Energize system facilitates development and understanding of C++ and C code on Sun SPARC and compatible workstations by incorporating capabilities previously available only in specialized OOP languages. The Energize Programming System is designed to speed the edit-compile-debug cycle by making possible incremental compilation and linking at the function and class levels. This ability to work at a finer level of granularity eliminates the need for programmers to wait through long recompilations and relinking, which is especially significant when only a small portion of a program has been changed. Also, because the system incorporates a production-quality C++ and C compiler, the programmer is always acting on the real program code and not a simulated version. Energize is composed of the Server that provides both control and data integration, and the Tool Set, which includes a suite of advanced new programming tools as well as familiar UNIX tools. Underlying the server, to store all the information collected about the program, is a commercial object-oriented database. Energize will ship on SPARC systems in Q1 of 1992 and will be ported to other leading UNIX workstation platforms over the next 12 months.

Send Product Announcements to Robert Shelton, 1850 Union St., Ste. 1548, San Francisco, CA 94123. Fax: (415) 928-3036. Include company name, address, and phone number.

Operating Systems

...The value of hardware brand names has dropped rapidly over the last few years, although Compaq and IBM and some other vendors retain enough cachet to command slight premiums. This is what I predict will happen to operating systems someday. Yes, the first vendors to market with the new operating system will have a slight advantage—just as the first vendors with a 486-based PC did—but it will be a fleeting edge. The reason has to do with the technical nature of objects and with customers' pressure for "openness."...Maybe it won't all be quite so simple or elegant. There will be leaders and followers, just as in hardware. But just as in hardware, the basic business is likely to be extremely competitive and fragmented. Who wins? Hardware and now operating systems will be competitive businesses. Yes, you'll probably get decent margins if you do a great job (and you'll go out of business if you're careless). You just won't get to be a Microsoft—not even if you start out as Microsoft. The real winners are likely to be the systems integrators, who will know what to do with all this stuff. You can be sure they'll charge the customers a hefty price for their wisdom. But if they can put all these modules together to deliver useful systems, customers will be happy to pay.

Snap-in software, Esther Dyson, ***Forbes****, 11/11/91*

..."Look at ACE—it's MIPS and Intel," [Gordon Moore, chairman of the board at Intel] says, noting that ACE has chosen Intel x86 architecture and the MIPS approach as the two hardware roads to the future. "This Pink thing...is IBM and Intel," he continues, referring to the plans of IBM Crop. and Apple Computer Inc. to codevelop an object-oriented operating system-code-named Pink—to run on both the Macintosh and IBM's x86-based PCs. "Sparc's Solaris group is now porting its software onto our architecture. So everyone out there sees the advantage to being on our base. We understand why we are a target for everyone to shoot at."

Waging war on Intel: the RISC crowd moves in, Samuel Weber, ***Electronics****, 11/91*

Antihype

...One of the criticisms I have of Turbo Vision is that it has no easily graspable set of Big Principles—you might say it's a collection of exceptions that totally overwhelms its rules. This is especially evident when you look closely at TV's family of views and how they have to be used. Some views need to be subclassed to use them, and some may be used as is. Some views must NOT be subclassed—and there's really no way to keep it all straight except to remember a whole raft of special rules...

Structured programming: the tragedy of the black box, Jeff Duntemann, ***Dr. Dobb's Journal****, 12/91*

...Objects are here to stay, and I'm glad. But I'd like to utter a warning: The hype and jargon now being slung around over object-oriented languages is exactly like the stuff that brought AI into disrepute in the 1980s. As my English teacher once said, "If you can express your ideas in clear, consistent language, they will live on after you." C++ code is revolutionary due to its very powerful implementations of polymorphism, inheritance, encapsulation, and overloading. C++ code is flexible and reusable. Which statement will still ring true in 2001?

The OOP wave, Michael R. Will, ***BYTE****, 12/91*

Databases

As part of their joint efforts to evolve object-oriented technology, Sun Microsystems and Hewlett-Packard are working with a bevy of object-database companies so that software developers will be able to write programs that will work with any object database. The database companies concerned are Ontos, Versant, Object Design and Objectivity, and the work will result in the development of a common applications programming interface known as an object database adaptor. The group, formed because of a lack of interest from the standards bodies, has met four times so far and plans to offer its completed work to the Object Management Group. The object database adaptor sits between the database and the object request broker, and when it receives a request for an object, invokes the corresponding database...

Sun, HP work on common object database access, ***UNIGRAM X****, 12/2/91*

...Still, there is no basis for a sufficiently detailed standard so that OODBMS applications can be developed in one system and moved with reasonable ease to another, such as an Oracle-to-Sybase conversion. The situation is more like an IMS-to-IDMS conversion. In other words, the differences between existing OODBMSs are more like the differences among structured systems (such as IMS, IDMS, and Total) of a generation ago than they are like the differences between relational products. Of course, there are standards groups working to rectify this problem, but the results, let alone the practical product implications, are still far in the future. Second, there is no theoretical underpinning to match the mathematical basis of the relational model. Consequently, design and implementation of applications is on a more ad hoc basis. The quality of an application is more dependent of the developer's master of an OODBMS than of a body of guiding principles...

...Both groups agree on a central core of ideas. But the OODBMS folks do not find the points about data independence and languages the relational guys make compelling because they don't help solve the problem the OODBMS people set out to solve. They care about the points, and they are embracing some features that make either technical or marketing sense, but they are not going to compromise their goals for someone else's views on technical purity. Conversely, the RDBMS experts are skeptical of the direction the OODBMS experts want to take database management because their experiences in database applications have shown them potential problems. And while RDBMSs are starting to include elements of object orientation, RDBMS people insist on also using what appear to many OODBMS application developers as unnecessary frills that will do nothing more than slow things down...

Database foundations: relational vs. object-oriented, Herb Edelstein, ***DBMS****, 11/91*

demonstrate that the conceptual approach described by the object paradigm itself has built-in productivity gains for application implementation and maintenance. The General Motors application, in contrast, is written in Smalltalk and uses an object database management system.

O-O is a concept, not a language or any particular set of tools.

SMALLTALK, C++, AND THE ALSO RANS.

The debate regarding C++ and Smalltalk is an interesting one for someone like myself who, in prehistoric times, was actively involved in both the PL/I and APL wars. It was then that I learned that emotional and religious conviction have as much to do with the choice of an applications language as functional capability or ease of use.

Clearly, C++ has its place. If a shop is staffed with C programmers, C++ is the obvious direction to go for much of their development needs. Experience has shown, however, that careful management attention needs to be paid to ensure that you do not wind up with extended C programs rather than object-oriented programs. C++ does not compel nor even necessarily encourage adherence to the object paradigm.

Converting the COBOL or PL/I programmer to C++ is a different matter. I won't say that it can't be done, but it certainly is a challenge. Several companies have demonstrated that it is much easier to induce a COBOL programmer to embrace Smalltalk than C or C++. That is not to imply that programming in Smalltalk is a piece of cake. It is not. But it is a little bit less arcane than C or C++, which are essentially systems programming languages better suited to the scientific, engineering, and technical programming communities.

The proprietary O-O languages are the also rans. Eiffel, Actor, even Objective C, have found niches; but they just don't bring enough differentiation to the table to ensure their acceptance by the MIS community.

O-O COBOL, recently announced by Microfocus, might gain acceptance by some commercial shops after it has had a chance to settle down. But, contrary to the publicity of Borland and Microsoft, the reality is that today MIS organizations such as American Airlines, Texaco, Citibank, and General Motors are using Smalltalk, not C++.

VISUAL PROGRAMMING

The technological advance that will attract the MIS community to O-O more than any other is the visual programming environment that obviates the need for procedural language programming as we know it today. This kind of visual object-oriented development environment would allow even sophisticated applications to be built by picking reusable objects from a palette and wiring them together to produce the desired functionality.

Servio Corporations GeODE object development environment implements this strategy. It also allows the wrapping of existing components such as bridges to relational databases. Legacy programs, e.g., existing image processing components written in C, can be manipulated as visual building blocks just as if they were specially written objects.

This kind of technology will allow commercial applications programmers, read this as COBOL programmers, to convert to O-O without facing the C++ or Smalltalk learning curve.

> ***"Fads come and go in MIS, just as they do in women's fashions."***

MIS WILL BE DIFFICULT TO CONVINCE, AND IT'S UP TO US

The acceptance of new technology by the MIS community would be the subject of an interesting sociological study in its own right. Like any group of communicating human beings, there is a tendency towards lemming-like behavior. This is compounded by the high costs involved in MIS, the poor track record of results in terms of systems that deliver exactly what the pointy end of the business needs, and the increasing complexity of the tools of the trade.

Fads come and go in MIS, just as they do in women's fashions. Sometimes the fad of the moment has been a productive technology, for example, the client/server model. In other cases, it has been an expensive sinkhole like Artificial Intelligence . What we have to hope for is that the reaction against a fad does not cause a valuable technology such as O-O to be passed over.

One of the inhibitors to the acceptance of relational database management systems in the early 1980s was the insistence by the database people on the use of terms such as tuple, normalized form, join, and so forth. It may have made them feel very sophisticated, but it took people a couple of years to discover that a relational database was a collection of tables. Once that became clear, it seemed a sensible technology to deploy.

O-O proponents have fallen into the same trap. A recent quote from Borland CEO Phillipe Kahn insisted

that to understand O-O one needed to understand the concepts of encapsulation, inheritance, and polymorphism. If the decision makers in MIS need to understand concepts with names such as these, forget it—O-O will never fly.

The concepts that make O-O relevant to MIS are neither complex nor new. It's just the way they are strung together that is different, and then only slightly so. The responsibility lies with us, the vendors, to ensure that we get a clear, simple message through to MIS management about object technology.

MAKING IT PAY OFF

As I said earlier, object-oriented technology promises to reduce the applications gap. I use the word promise, not the word guarantee.

With the debacles of AI, and to a lesser extent, structured programming and CASE behind us, it should be pretty obvious that there are no silver bullets in this business. O-O is just as far from being a silver bullet as the previous technologies. It has more promise, but the challenge of managing it will be no less.

Listening to the small number of people who have already successfully deployed O-O in the MIS world, there is a consistent theme in their reports: Don't expect too much return up front. Building reusable objects is not easy; it takes skill and experience and it may take three tries to get the object right. The subsequent benefits provide the payoff: you can reuse the object until the cows come home, repackage your application in hours instead of days, make minor changes or corrections in minutes.

Unfortunately, the design of reusable objects has to be carefully thought out, the implementation needs to be very good, the testing comprehensive, and the documentation complete. There is no substitute for discipline in the development process. If anything, it is more important than ever in an O-O environment.

THE CULTURE TRAP: THE FUTURE IS CROSS CULTURAL

Having overcome other barriers related to persuading MIS management to deploy O-O, there will be a remaining problem that greatly worries me. There is a culture gap between the computer scientists and the MIS community. By computer scientists, I mean those who are trained in languages such as C or Modula, and systems such as UNIX, rather than COBOL and MVS.

The crucial problem is not the limits of knowledge of either group: it is the attitude of one group towards the other. The immediate future lies in marrying these two worlds together.

To successfully implement object-oriented technology in MIS organizations, we need access to the best skill and experience on each side of the cultural divide. How can we best integrate and divide function between the desktop workstation and the mainframe? If you are not intimately familiar with the technology of both the workstation and the mainframe, you cannot contribute satisfactorily to the development of the best solution to the problem. If you despise the technology of one side or the other, and its practitioners, no reasonable solution can emerge.

Those who are currently involved with object technology are, in my categorization, as you might suspect, computer scientists. There is a tendency among them to disparage COBOL and mainframes, just as the UNIX people do. We have to get away from this. You cannot expect to win hearts and minds with a slap in the face. It has to be done by understanding the problems of the MIS community, its concerns and misunderstandings, and working with them to bring O-O to bear on problems.

ALL THE INGREDIENTS ARE IN THE POT

Right now, everything we need for the successful deployment of object technology in the MIS world is ready and waiting. The trauma that the computer industry is experiencing at the moment has actually improved the chances that O-O will be deployed by MIS. People are taking stock of their successes and failures with applications and systems. They are looking for technologies that will help them avoid the mistakes of the past. O-O is it. Let's make a major effort to ensure that they don't screw it up!

He who proposeth a technology,
Hath responsibility to keep it simple
And maketh work..
— O'Shea ≡≡

Donal O'Shea is President of Servio Corporation, provider of the GemStone object database management system and visual development tools.

Applications

...The University of Colorado at Boulder's Center for Advanced Decision Support for Water and Environmental Systems (CADSWES) is developing an object-oriented river basin decision support environment that lets users assemble river basins to model the effects of policy options. This system, River Simulation System/Multi-Objective Decision Making (RSS/MODM), is being developed in cooperation with the US Bureau of Reclamation. Although the initial test case for the system is the Colorado River, its architecture lets it be employed on any river system....The CADSWES development team chose Franz's Allegro Common LISP as the basis for prototyping the system software. Since this project is trying new ideas in the field of water resources, LISP's ability to prototype rapidly was essential to the development process. The use of LISP also let them incorporate previously conducted research work into the project. However, the final implementation will be in C++ with MOTIF as the graphics standard...

Application watch: down by the lazy river, Patty Enrado, ***AI Expert****, 12/91*

Methods

...Requirement views, context specification, event-object partitioning, environment modeling, design templates, method hierarchies, and a rapid system development strategy are all part of a comprehensive object-oriented approach to system development. All these elements evolved from structured techniques into a significant number of today's O-O methods. These methods in various forms have been used and written about extensively for the last ten years. Other concepts required by object-oriented development are readily available from entity-relationship analysis, Jackson's data-structure oriented approaches, and the Smalltalk language. These sources for object-oriented methods have been around for a long time as well. As with elements of the structured techniques, method developers have been incorporating those ideas into what are now known as O-O methods. This is not to say that every last issue in object-oriented development methods has been resolved, but the evolution to revolution question is rendered moot given that the structured techniques have already spawned O-O methods. Instead of revolution, the real issue is revelation. For a variety of reasons, many in both the object and the structured world have slept through the evolution of O-O methods. For all of them, I hope this article is a wake-up call.

Debate: evolution vs revolution: should structured methods be objectified?, John Palmer, ***Object Magazine****, 11-12/92*

...The notion of assertion is the basis for what I view as the key component of the methodology underlying O-O software development: the theory of design by contract, which views software construction as the design and enforcement of successive contracts, large and small, between clients and suppliers, each party keenly aware of both its rights and its obligation. The contract theory makes it possible to base the construction of reliable software systems not on informal hopes reinforced by careful testing, but on formally specified relations between cooperating software elements. It also helps to understand the nontrivial aspects of object technology and use them properly. In particular, exception handling becomes the job of getting out of a contract in a safe way when the other party does not observe its part of the job; as to inheritance and dynamic binding, they provide the necessary subcontracting mechanism...

Debate: evolution vs revolution: should structured methods be objectified?, Bertrand Meyer, ***Object Magazine****, 11-12/92*

Environment

...Actor has always had the misfortune of being pigeon-holed as a hard-core tool, one that required more work than environments like Astymetrix's ToolBook or Knowledge Garden's KnowledgePro Windows. It has been accused of placing too many layers between Windows and the developer. But at the price of some incompatibilities with previous versions of Actor, the addition of Action methods, WDL, SQL library, and protocols establishes Actor as a complete, mature product that is robust enough to consider as a first-choice Windows-based, object-oriented environment.

First looks: Actor 4.0 matures with safe inheritance, SQL classes, and graphics, Richard Hale Shaw, ***PC Magazine****, 12/31/91*

Languages

...Hewlett-Packard has an in-house group dedicated to implementing software reuse. Unfortunately, they have made the mistake of trying to implement the reuse policy using a low-level language such as C. From my point-of-view, C is simply a universal assembler and doesn't lend itself to the development of truly reusable software. Instead, HP should be using a language such as C++ or Ada to meet the "without modification" goal of reusable software. Many see the advent of object-oriented methods as an opportunity to make reusable software components a reality...

...Ada is currently regarded as an object-based language because it does not currently support a full inheritance mechanism. It does, however, support the fundamental goals and principles of software engineering, and does support limited inheritance. C++ supports inheritance, but falls short in...[the] development of generic components. The C language is highly flexible but poorly suited to engineering large, reliable software applications...

What we mean when we talk about software engineering: objectively speaking, Richard Riehle, ***HP Professional****, 11/91*

F.Y.I.

Excerpts from leading industry publications on aspects of object technology

CASE

...Developers using object CASE tools without the prerequisite training, practice, and a fundamental understanding of objects will face the same failure as the inexperienced doctor operating on patients using state of the art equipment. Tools alone cannot save the patient!

Point/counterpoint: using CASE tools without the methodology, the developer's perspective, Kathleen Meyer, ***Object Magazine****, 11-12/92*

...Revivalist ministers, professional pontificators, famous name endorsements with little or no content contributions, and famous corporation affiliations cannot be the criteria by which your organization selects an approach to object analysis and design. There are a limited number of experts currently on the object market, and there are even fewer analysis and design experts in any market. This is an unfortunate but true statement as evidenced by the all too frequent question, "Will object technology replace CASE?"...CASE is not a technological concept, it is an enabling technology. It just so happened that most of the CASE tools in the early/mid-1980s were built to support the notation of structured analysis and design....

So, what really is this editorial about? It is a strong admonition not to view an object analysis and design book, technique, or CASE tool as a means to learn and implement object technology—no matter what the marketeers try to sell. Just as objects are not the silver bullet for software development, object analysis, design, CASE and methodology are not the silver bullet for gaining knowledge and expertise in object technology—particularly now when the techniques and tools are still being conceived of.

From the editor, Marie A. Lenzi, ***Object Magazine****, 11-12/92*

OOP

..."[object-oriented programming is] the difference between building a castle out of sand and building a castle with Legos," says Roger Heinen, Apple's vice president of Macintosh Software Architecture...

Software industry is "object-oriented," Rory J. O'Connor, ***Huntsville Times****, 11/10/91*

...But so far the move to object-oriented programming has been disjointed because of the lack of standards for the vocabulary, syntax, and processes that make up the object architecture. Object-Oriented Design with Applications, by Grady Booch, breaks new ground in setting forth a discipline for effectively applying the elements of the object model. Booch provides a fundamental explanation and example of the object model for a wide audience of software engineers, analysts, system architects, project managers, and students....The book's only serious flaw is its tone; too often it reads like a textbook...

Read Only: comprehensive "how-to" manual leads the way to object- oriented programming, Bob Airasian, ***PC Magazine****, 12/17/91*

...As far as object-oriented technology goes, [Bob] Libutti [programming systems director of market strategy with IBM] said that IBM was considering using Sapiens tools internally in several industry areas. And prior to its discussions with Sapiens it has been talking to Digitalk. Indeed, SmallTalk is used extensively in IBM—it was used to build the new Cross System Product, CSP, and was also used by Intersolv Inc. to build its product line for OS/2. Libutti said that IBM is making a major investment in object-oriented technology. This is all well and good except that IBM's Repository is based on the Entity-Relationship model and not the object-oriented model. However, Libutti says that IBM is extending the Repository to cope with object types, although when pressed said he didn't know when these extensions will appear...

IBM sets the record straight on misperceptions of systems application architecture, ***UNIGRAM X****, 11/18/91*

...How should analysis and design be used with object technology? The answer is: incrementally, in conjunction with actual development, using separate techniques for each layer of a layered system architecture....Instead of designing an entire monolithic system on paper before beginning implementation, sketch out a skeleton system to get you started, then code that much and see how it works. What you learn from the implementation is bound to improve the design. Then take that improved design, sketch out an incremental set of functionality, and roll that into the code. The malleability of object technology makes this incremental approach to development far more feasible that it was with conventional technology. In addition, tons of paperwork can be reduced to mere pounds, the product rolls out faster, and the overall design is usually superior because it is proven in the field and enhanced in stages rather that being thrust upon end users in the conventional "big bang" fashion...

Easing into objects: redefining analysis & design, David A. Taylor, ***Object Magazine****, 11-12/92*

Multimedia

...Multimedia personal computers—which combine stereo sound, sophisticated graphics and motion-picture-quality video—is the computer technology likely to have the biggest impact on the industry next decade, according to a poll of attendees at last month's Comdex show in Las Vegas. The poll, commissioned by Byte Magazine, ranked multimedia PCs first, followed by portable computers, wireless communications, and object-oriented programming....One interesting finding: While Comdex this year was populated with companies introducing penbased PCs, only 10% of those polled by Byte thought pen computing...would be a major development...

Techtalk, John Schneidawind, ***USA Today****, 11/26/91*

RETROSPECTIVE

1991—The year it all changed!

by Tom Love

Who would have ever thought the USSR and IBM would be dismantled in a single year or that IBM and Apple would team up to build an object-oriented operating system?

Or that IBM would become a reseller of Smalltalk/V and that object-oriented technology would be featured on the cover of *Business Week?* And who would have projected that 500,000 copies of Borland's C++ could be sold?

What a year! There was just so much to cover. I want to begin this retrospective with a variety of observations, then I'll try to make an assessment of the implications of these events, closing with two predictions for the future.

OBSERVATIONS

Some deals and events in our industry deserve special notice:

- Symantec acquired Zortech in June.
- Gemini invested in ParcPlace in September.
- Sequent, Versant and ParcPlace have partnered to deliver a parallelized version of Smalltalk coupled with an ODBMS.
- Momenta delivered on its promise to provide a pen-based computer whose applications are all written in Smalltalk.
- OMG members are in apparent agreement on the Object Request Broker (ORB) standard.
- General Electric started teaching a suite of object technology courses.

This has been a year of major challenges for our industry as well, such as:

- The major business slump at Mentor Graphics was caused in part by its "bet the company" decision to use objects on the largest project to date anywhere—over 12,000 classes in C++.
- The fierce slugfest continues among object-oriented database companies to achieve product dominance and company survival in a huge and potentially lucrative market.
- Object-oriented COBOL seems *stuck in the birth canal* while potential customers look elsewhere for an answer.
- KnowledgeWare imploded, triggered by the failed romance with IntelliCorp.
- The average experience of object technology trainers was reduced by half again this year due to massive demands to train increasing numbers of people.

Some other observations are remarkable in their own right for what they imply about the maturity of object-technology:

- There was a discussion about object-oriented programming on the Larry King Live television program.
- A full bookcase is now devoted to objects at Silicon Valley's famous Computer Literacy Bookstore.
- Venture capitalists are beginning to see success stories with objects and with Smalltalk and no longer shun products built "the right way."
- Someone has actually built a batch version of Smalltalk that runs on MVS for use in building commercial business applications—dyna-batch under glass!
- Application software houses like Aldus, Claris, and Symantec have belatedly gotten the message that their

Tom Love, PhD, is an independent management consultant.

products could benefit from object technology and are quickly scurrying to hire object-oriented programmers.

THE BUSINESS OF OBJECTS

A dose of reality about this business is beginning to be accepted. People are starting to stand up at conferences and say, "It takes a couple of years to become fluent in object technology." As a result, the demand for experienced object-oriented designers and developers is soaring, as are the prices for such people.

Though many large commercial organizations are poised to use objects, many of these organizations are also poised to fail. Too many people are rushing to apply objects on their next big "bet your company" project. Instead, they should plan to use objects on the next three projects and then plan accordingly.

As I look at the industry at the end of 1991, Borland has clearly racked up higher sales figures than anyone else. But SIGS Publications may have produced the highest return on investment. The highest losses still belong to the O-team, the object-oriented database companies. The most significant technical achievement belongs to Momenta.

Big time marketeers are beginning to circle this fledgling industry. Watch the ad budgets and the mail in your in-box grow over the next few years. Think about expanding the size of your in-box now, before the rush.

O-O DESIGN AND CASE

Methodologists are writing a lot of books. Some have even taken the time to write a few modest programs with objects. On the O-O design front, Rumbaugh, et al., has taken the deserved lead. Jacobsen keeps telling us how good it is going to be, and Coad's code is weaker than even Yourdon's name can overshadow.

I wish all of the CASE guys would realize that text-based inheritance trees are just fine. So are 3x5 cards. If you are not going to provide some real value with your tools, save your disk space for something useful.

New object-oriented programmers are writing even more books. I recently saw a bookshelf in a museum with a few books on C++ programming. I know not why.

O-O LOWER CASE

Dave Thomas continues his campaign to make team programming a reality for Smalltalkers. His company's Envy product is the envy of the eye of every Smalltalk project manager to whom I have spoken. It is also just the kind of tool that we must be building to march this technology into the industry mainstream.

Unfortunately, we have yet to find the Dave Thomas who wants to take on the "testing problem." Somebody has to solve that problem this year; it is not that hard!

There are two observable trends in the C++ community. There are several new commercial products shipping that use the language successfully. For others (especially those who have written anything in Smalltalk or even Objective-C), the disenchantment grows. Active hostility is sometimes observed in public, such as during the Apple-IBM announcement at OOPSLA.

Nevertheless, C++ compilers are selling remarkably well, though there are other indications that these same compilers are getting very little exercise. One recent survey showed that only one in twenty purchasers of C++ compilers were using the product. There is no doubt that the complexity of the language limits its acceptance. Users continue to chant, "But it is the de facto standard," as they walk lemming like to the coffee machine while waiting for a compile to complete.

Objects can provide real value. A complex, hybrid language like C++ has either got to get fixed or it is going to evaporate. "Fixed" means we should stop thinking of what to add to the language and instead think of how much we could take away and still have something of real value. Remember: Multics had a lot of theoretical power, but its daunting complexity caused it to be remembered only as the big project that spawned UNIX.

Maybe the real genius of Philippe Kahn is that he figured out how to achieve massive sales numbers with little or no need for ongoing support or maintenance. Time to sell, Philippe.

THE CLASS LIBRARY BUSINESS

It is still the case that too few class library products are under development and even fewer are in the market—less than 50 by my count. Trust me guys, this is where the money is. Too many people still believe that class libraries and GUIs are synonyms. We have enough string classes, collection classes, and menu classes now. Move on to something challenging that other people do not want to build themselves.

Regrettably, the quality of class libraries has not become much better. Purchasers of class libraries are almost always purchasing untested code. I recommend only paying for classes when you see the test cases.

EDUCATION AND TRAINING

A limiting factor of the industry is the rate at which educational institutions are training students in O-O languages, design methods, and class libraries. Recent discussions with university recruiters revealed the staggering finding that most computer science grads have not heard of objects. Where have they been and what do they read?

continued on page 12...

and receive faxes while the user is writing on the screen. For resolution and character recognition, the pen itself requires 100-200 events per second, compared to a mouse that requires 25-30. We use a specialized I/O processor to off load the main cpu.

HOTLINE: Would you do anything differently in light of what you know today?

INFANTE: I can't comment in detail at this point. There is other technology, and today we would like to be able to take advantage of new productivity tools. We might still come to the same solution. Every few years, you need to reevaluate your technology and see if there are minor or major changes that could make your product development more effective. Right now, we're at a convenient breaking point having shipped the first product. We are now reevaluating our process to see what we should do differently for the next generation. Continuous improvement requires periodic reevaluation of your processes to see what can be done better.

HOTLINE: Were there any particularly good or rough experiences? Challenges?

INFANTE: Learning the technology, learning how to manage the development process. The structure of the development team has to be different with object technology. An early article in *Hotline* on "Objects by Teamwork" was very illuminating, but at the time there were few companies with a lot of experience delivering shrink-wrapped products to the outside world using object technology. Managing development and communicating to the rest of the company what was happening was a challenge. With Smalltalk, we could prototype quickly, but we had to restrain people from going out and selling the prototype. It was a prototype, not a shippable product. This was a management of expectations problem. In a less supportive management environment where the corporate management team did not have confidence in the development team, expectations could easily have gotten out of hand. The other side of being able to demonstrate a prototype quickly is that when you're developing real product, nothing works for a comparatively long time. Then, everything suddenly works. It's steep cliff that you must manage very carefully.

HOTLINE: Could you summarize your experience bringing the pentop to market?

INFANTE: It's been incredible fun. We've accomplished more in the last year and a half than most companies accomplish in a lifetime. When I think that we had our first Smalltalk training class in April 1990, and a team of a dozen people developed a complete operating system and full application suite in 18 months, starting as good business people and not language people, it's impressive. That includes software and hardware development, and product integration. ≡≡

HOT OFF THE PRESS

MIS managers and developers: Object-oriented COBOL takes the next step toward availabilty as a standard, commercial product.

The Object-Oriented COBOL Task Group met December 9–13, 1991, in Palo Alto, CA. Members of the task group are IBM, Hewlett-Packard, Hitachi, Micro Focus, SPC Systems, and Realia, with many other companies participating as observers.

At the meeting, the task group agreed to a feature set that will constitute a Version One O-O COBOL. This recommendation is a key step toward making O-O COBOL available to users.

As this issue of the Hotline goes to press, the next step in the process is taking place. A Task Group presentation of Version One to a joint meeting of the ANSI X3J4 COBOL Committee and the CODASYL COBOL Committee took place January 13–17, 1992. This is an important milestone on the path toward achieving an O-O COBOL standard.

In the March 1992 Hotline, we will cover the contents of the Task Group's recommendation and the results of the ANSI/CODASYL presentation. This should give COBOL users an idea of what to expect from early releases of O-O COBOL.

—Editor

HOTLINE: You've mentioned design methods several times. What disciplines and tools have you used in designing the pentop environment and applications? How rigorously did you apply them?

INFANTE: We started with CRC because that's what KSC taught us. We used 3x5 cards. We've been unrigorous. Initially, CRC was used heavily because we felt unsure of our design skills. We have not gone back to CRC. Today most of the design is done on blackboard and with written documents. We use a word-based description of the overall design, then a methods description. We don't use diagramming techniques, and have not found the need to use things like state diagrams. We wrote a specification for many parts of the product, although we were fully prepared to iterate on the user interface component. We wanted Smalltalk to enable us to change the product easily. We expected to need to iterate many times on the user interface (UI)—not once or twice, but six or seven times as we tried out our product on focus groups to get customer feedback. After user testing with prototypes, we threw some designs away because they didn't work. But we learned from them and got progressively better. The user interface, in particular, needed to be compelling and compatible.

> ***Our objective was to provide a systems solution usable and complete out of the box.***

HOTLINE: Momenta seems to have taken the same direction we're seeing quite often in this industry. You teamed up with specialist providers to deliver an integrated solution. How did this work?

INFANTE: We faced up early to the fact that for the first year or two, we were not going to have a whole bunch of developers (ISVs) for our product. Ours was a systems project. That was a factor in selecting Smalltalk and an O-O design methodology because we wanted to develop a complete office/desktop solution for the customer—word processing, spreadsheet, graphics presentation package, and fax. We needed to develop a number of them in-house because we didn't expect it would be feasible to recruit a large number of ISVs right off the bat. We focused on three to four ISVs who are key and committed, and who will work closely with us. We bundled those with the box as a complete solution. The unit is useful out of the box—unpack it, and turn it on. That's all. The software is already loaded. The automatic boot-up routine helps users fax in their warranty card. Our objective was to provide a systems solution usable and complete out of the box. This also included file transfer/connectivity, modem software so the user can upload software upgrades, a Xircom parallel multiplexer for attaching to a Novell network, interfaces with smart cable and cellular, a calendar, an address book, etc.

HOTLINE: What development tools did you use? What about commercial class libraries? What worked and what didn't?

INFANTE: We were fairly limited in what could be acquired. There wasn't a lot of ready-made stuff out there at the time that we could appropriate. Envy Developer from OTI was one of the biggest factors in our success. It allowed us to do team development, and was an order of magnitude better than any other source code management system I've seen. We also used Acumen, the widgets package. We bought it and essentially completely rewrote it. We did not feel we were sufficiently expert in designing Smalltalk to up with an applications framework from scratch when we started. We studied Acumen until we felt we could surpass it, then we did something completely different. The package definitely bought us time to market.

HOTLINE: One prejudice I often hear expressed is that Smalltalk is not fast enough for time-critical applications. I often hear programmers promote C++ for "performance reasons." Yet, your application seems more like a real-time application than many on which C++ has been used for "performance reasons." Is performance a problem here?

INFANTE: One of the criteria in selecting Smalltalk was performance while running a full applications suite, including real-time services. The main reason we decided against C++ was that, at the time, the people who were in the process of delivering C++ based products were not able to get to market on time. We needed high performance throughout the development cycle as well as acceptable runtime performance. C++ projects were having trouble delivering the application in any performance state. As for our product, we would like to be at least 50% faster, but we have found that customers are excited enough about the product and advantages that they are willing to live with the performance for a few months. We know where the bottlenecks are in our designs and code, and can fix them with engineering work over the next few months. It's not a language issue. The system is a real-time application. It can print

Technology Transfer ≡≡

Making the transition to object-oriented technology

by Ron Suarez, PhD

During a recent visit to a major telecommunications company, I had the opportunity to discuss the organization's experiences with making the transition to object-oriented technology.

What I heard highlights some of the obstacles that companies often encounter when making the change to object technology.

My first impression was that the manager who was assigned to help make the transition did not see the value in what he was doing. He kept questioning me as to why object-oriented technology would make such a difference. There seemed to be strong undercurrents of resistance to the coming changes. I sensed that someone in upper management had said, "Use C++," and the subordinates were being dragged kicking and screaming into the world of objects. In-house training in C++ had been conducted, but the software development managers did not grasp what was important or different about objects. They simply had been told that C++ would make their developers more productive.

Ironically, on my way to this meeting I had an experience that Carl Jung would call synchronicity. I had stopped for something to eat, and at the table across from me a software consultant was negotiating a contract with a prospective client. "The first thing we will do is a requirements analysis," he said. "During this stage, we will define all the requirements of the system. Then we can write the software specifications." The client responded, "What if we don't think of everything we need during the requirements analysis stage, can we make changes later on?" "Sure," responded the consultant, "but those will have to be kept to a minimum."

This overheard conversation about the traditional software development process foreshadowed my discussion with the telecommunications company managers. One manager said, "We have about 40 people conducting requirements analysis and we would like to have them continue doing their job. What do you think about continuing that approach, but using C++ during the implementation stage?" He wasn't very happy with my response. The overwhelming consensus of people doing object-oriented development indicates that, if anything, the opposite approach should be taken. Development teams are better off adopting object-oriented analysis first, even if they continue to code in a language that is not object-oriented. I explained that the task of making the transition was much more involved than simply learning a new language. The employees would have to undergo what we call a paradigm shift. They would be learning an entirely new way to conceptualize the problems that they were asked to solve.

In this article, I will elaborate on the need for these deeper changes, and explain why obtaining productivity benefits from object technology depends on adopting object-oriented concepts in the analysis and design phases of application development. Changing languages is not the key to the benefits of object technology.

There are risks involved when making such a shift in thinking. Many of them are rooted in organizational behavior issues that cannot be ignored. Attempts to produce major changes in team and individual behavior need to be planned carefully. Organizations cannot simply focus on technology. Employes may fear losing their jobs if they cannot make the expected shift. Resentment may produce exactly the opposite of the intended productivity gains.

While organizational behavior is not the focus of this article, we need to be aware of the risks, people's fears, and ways of working around the problems. One approach

Ron Suarez, PhD, is President of Arbor Intelligent Systems, Inc., a provider of training, consultants, and Smalltalk-based products for object-oriented development.

is to ease the transition by initially working with smaller numbers of people, gradually bringing others into the process. In addition, employees learning object technology do not need to start using C++ or Smalltalk immediately, or for all their work. What is critical, however, is that they learn to incorporate object-oriented principles into their analysis and design. Cognitive psychologists tell us that the best way to learn new behavior is by active participation. I will describe and give references for approaches that can be used to help people begin to actively use object-oriented analysis and design.

OBJECT-ORIENTED DEVELOPMENT WITHOUT AN OBJECT-ORIENTED LANGUAGE

In our work at Arbor Intelligent Systems, we usually use Smalltalk and C++, but sometimes we still need standard ANSI C. David Feldt, who conducts training and consults with our company, maintains that the object-oriented approach to software development does not require the use of an object-oriented programming language. Basic object-oriented concepts like encapsulation, polymorphism, and inheritance can be achieved in traditional programming languages using existing language syntax.

Good examples of this can be found in commercial software libraries and development systems. An increasing number of commercial products are designed with object-oriented concepts in mind, but they are coded in nonobject languages. These products require knowledge of object concepts on the part of the programmers to be used effectively. An example of such a product is Neuron Data's Open Interface. Open Interface is a standardized graphical user interface (GUI) library that supports the creation and maintenance of code on Macintosh, Windows, Presentation Manager, OpenLook, and Motif. Although written in standard ANSI C for portability between platforms, the structure of the library is heavily object-oriented. Users unfamiliar with the concept of inheritance will find the architecture and use of this library confusing, while those knowledgeable in basic object-oriented concepts are better able to make immediate and effective use of this product.

David's experience teaching programmers who have struggled through the transition to object-oriented development suggests that there are important concepts in traditional software engineering that can help developers. These include disciplined exception handling, coding style, and naming conventions and rapid prototyping. David also points out that the culture shock experienced by the professional programmer when first confronted with objects must not be underestimated. The time required to become comfortable with object concepts, and to develop new work habits and thinking patterns, can be substantial. It is important to begin educating developers about object concepts now and begin using object-oriented methods that can assist developers in becoming more effective and productive in light of these changes.

ONE WAY OF USING OBJECT-ORIENTED THINKING THAT WORKS FOR ANALYSIS AND DESIGN

One technique that has proven effective in technology transfer and on actual development projects makes use of the class responsibility collaborator (CRC) model of analysis and design. This approach emphasizes thinking about the real life relationships between objects and defers thinking about the representation of data until it can no longer interfere with analysis and design. A lesson learned from using CRC is that developers often react more favorably to object concepts when offered practical techniques rather than being told in some abstract or academic way that everything they know is wrong. Use of practical techniques like CRC can yield substantial benefits even in the absence of an object-oriented language or development system.

Kent Beck and Ward Cunningham[1] have described their use of CRC cards. This method has also been elaborated on by Elizabeth Gibson.[2] Beck and Cunningham have defined three dimensions that identify the role of an object in a system being designed: class name, class responsibilities, and collaborators. They have taught introductory classes using these CRC cards to get people to start thinking in an object-oriented way. An example they use is an automatic banking machine. In this example, two classes would be `Account` and `Transaction`. Many people new to object technology have difficulty with this first step—deciding whether something should be a class. Elizabeth Gibson recommends that you first interview the users of a prospective application. See what they do, who and what they interact with, and what the outcomes are of different actions. The results will be a list of desired and necessary behaviors for a system. Classes become more evident when you ask who or what conducts groups of these behaviors. In the CRC approach, class defines who or what is responsible for a particular behavior. Beck and Cunningham use 4 x 6 index cards to write all the information for a class. Figure 1 shows that for an automated banking machine the class `Transaction` has two responsibilities: (1) validate and perform money transfer and (2) keep audit information. To carry out its responsibilities, it must interact with the following collaborators (other classes): `CardReader`, `Dispenser`, `RemoteDB`, `Action`, and `Account`.

A particularly salient aspect of Beck and Cunningham's experience in using CRC cards to teach object-oriented thinking is their reluctance to computerize the exercise. They see significant value in the developer's ability to physically move the cards around. Learners

INFANTE: We looked at a number of different options and ultimately settled on Smalltalk and an object-oriented design. Among the alternatives were programming under Windows 3.0 or the Windows pen version, when it became available, and traditional programming languages. We selected Digitalk Smalltalk\V286 in part because they offered a product that was very small. The basic product takes about 300K of memory, though we've added a few megabytes to that with our applications! At the time, Smalltalk was perceived as a fringe effort. We spent quite a bit of time evaluating whether it could be a viable delivery mechanism for shrink-wrapped applications. We were concerned about performance, memory utilization, and management of the programming environment. There was a whole list of criteria. [Ed: see box below.] We found that the reliable references were those who said, "You may get a ten-times development boost, but there's still testing and design time, and those are less malleable to O-O techniques." You still have to define and design the products.

HOTLINE: Are you saying that object technology and the Smalltalk language don't eliminate specification and design activities?

INFANTE: Smalltalk makes it easy to try to skip design and prototype your way to a solution, but that is not the right way. Architectural definition and detailed design are a requirement in any methodology and any language. Smalltalk is very seductive because you get stuff on the screen, and it does things. But you can't just ship it. You still have to define and design your product. And to get a quality product, you still have to test it. You can compress those tasks, but you can't make them go away and still get a quality, reliable, supportable product.

HOTLINE: So you entered the project with a team that was experienced in Smalltalk and understood how to specify and design a product using objects?

INFANTE: No. The team was not loaded with Smalltalk heavies. They were a group of C programmers. A couple of people had used Smalltalk for prototyping, but they had never delivered a product with it. We put the entire team, about seven to eight people at the time through a one-week Smalltalk course as a group. Knowledge Systems Corporation (KSC) taught that first class. Since then, our existing developers have taught the newcomers. Also, we have held in-house training classes for our ISVs, using our programmers to teach a combination of Smalltalk, design method, and the Momenta applications framework.

HOTLINE: What you're saying is you took procedural programmers, trained them from scratch in objects, and delivered an integrated product to market twenty months after you started? Wasn't the transition difficult? What about the learning curve?

INFANTE: The biggest difficulty we experienced was with bringing in the next four or five developers. They came in about a month after the first team had been trained. We were not really ready to train the newcomers because the existing programmers were not experts yet. The problem has been remedied because of growth of expertise, and now we have about fifteen developers programming in Smalltalk. Only one of our developers was a Smalltalk expert when hired. Now that we're established, we can typically take on high-caliber hires with an average of ten years experience in the industry, and it takes about one month for them to grasp O-O concepts. It can be frustrating at first because they have ten years of prolific C code experience and are used to hacking out sixty thousand lines of C a year. They have to get over writing Smalltalk that looks and behaves like C. You'll see relatively little progress for the first month or so. They understand the syntax at first, then about a month into it the light bulb comes on. Within another month or two they are proficient and expert. There is a steep learning curve that you must face unless you are willing to hire only established Smalltalk programmers, and those are hard to find. We looked for experienced developers of spreadsheet, email, and presentation packages; developers of large systems that had been shipping from large companies. We hired for the business skill and trained in the Smalltalk. It takes two to three months to get them as expert in Smalltalk as they were in their former language, but it's worth it for us because you get someone with expertise in the business you're in. That is more relevant than language knowledge. My attitude is, and this is partly the Momenta maxim, hire the best. Emphasize teamwork. If you hire somebody who has ten years experience, has shipped half a dozen products, and has a college education or equivalent experience, two to three months is a small investment to make someone a highly productive part of your team.

> ***Architectural definition and detailed design are a requirement in any language.***

and we wanted to extend the standard differently than the direction that Windows was going at the time [Ed: late 1989, when Windows 3.0 was in beta] while also providing a future path to DOS and Windows compatibility. We wanted to ultimately reach consumers who were not computer literate. We wanted to offer an environment that could downsize to a penpad. It had to be intuitive, user friendly, and utilize the pen fully without giving up the keyboard and DOS. It also had to be competitive with off-the-shelf applications.

HOTLINE: So the keyboard is essential even on a pentop?

INFANTE: The user must be able to use a keyboard. They're of tremendous value for data entry, taking interview notes, and writing articles. But to design a truly pen-centric product, you need to rethink the user interface from the ground up: how could you run the machine without complex interfaces, function keys, and a mouse. We designed the pen-centric interface first, then added the keyboard back in so that the whole thing would work smoothly with existing standards.

HOTLINE: The design objectives seem to reflect a tension between between existing PC standards and the completely new style of a pen interface.

INFANTE: Yes. We call that revolution with evolution—revolutionize existing standards in an evolutionary manner. You aren't selling into an empty factory. Customers won't give up what they have and know. Most companies have a PC or Mac. The pentop must be able to fit in even though it's very, very different. It was an incredible challenge. That's one of the reasons we chose object-oriented technology.

HOTLINE: So using object-oriented technology was a design goal?

INFANTE: No, object technology was not a design goal. Our goal was a PC-compatible pentop suitable for horizontal market white collar workers that came complete out of the box. Object technology was not an objective, it was a solution. We did not have a technology obsession. We view object technology as a very efficient way to reach our goals. We did not start out wanting to use object technology. It was "this is the kind of product we want to build, now how can we get there?" Time to market was a key factor in our selection.

HOTLINE: Could you tell us about your selection process? How did you select objects, Smalltalk, and ultimately Digitalk Smalltalk\V?

A summary of Momenta's selection requirements

Momenta addressed a significant number of detailed questions of existing users of Smalltalk prior to making its selection. Representative questions have been included here to help you develop your own criteria for evaluating development products.

- What are the vendor's primary strengths? Weaknesses?
- What is the overall quality of the product?
- Describe the level of support you receive? Cost? Response time?
- Would you use Smalltalk as an applications delivery vehicle for shrink-wrapped products?
- [If yes] What are some factors that might prevent you from using Smalltalk for such products?
- [If no] What would be required in order for you to use Smalltalk for such products?
- Assuming you are using Smalltalk now, what would you not use Smalltalk for?
- What are the productivity differences you see between Smalltalk and C? What are your measurement criteria?
- What criteria do you use to determine whether to program in Smalltalk or in another language? Within one project? From project to project?
- What is the size (in terms of code or classes) of the Smalltalk projects conducted in your organization?
- What is the biggest resource utilization problem you experience with Smalltalk? How do you get around it?
- What design constrains must your product meet (e.g., absolute code size, or specific processor families)?
- What is your design methodology? How has your design methodology and your organization changed as a result of object orientation?
- What proportion of your overall development is user interface?
- What importance do you place on user testing and prompt incorporation of user feedback into your product? How do you accomplish this?
- If you could change anything about Smalltalk, what would it be? Why?
- How did you handle training? Who did your third-party training and what services did they provide?
- What was the biggest problem in getting your organization up to speed with the product?
- What mechanisms do you use to ensure quality in the completed product?
- Would you use Smalltalk again or are you considering its use on a future product? Why or why not?

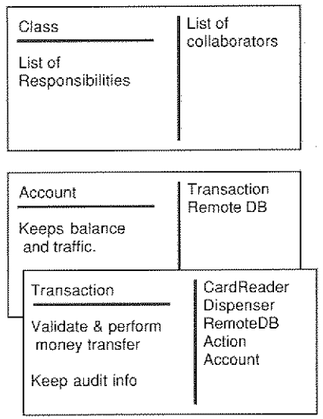


Figure 1. An example of using CRC cards for object-oriented design.

were better able to see the design from the differing perspective of a specific card in their hands in relation to varying arrangements of the remaining cards. Perhaps virtual reality interfaces of the future will provide three-dimensional interfaces where analysts may manipulate virtual cards. Until then, simple paper cards may be the most effective, low-cost tools for implementing this technique.

The goal of object-oriented methodologies has been to create a new system of representation that has a closer mapping to the real world of objects than structured analysis and functional decomposition. However, the attempts at presenting these new methodologies have too often mystified rather than clarified the approach. Beck and Cunningham have reported that, while many methodologies have left some programmers behind, this method has been effective even with weaker programmers. To learn new concepts, we should be able to associate them with previously understood material and actively manipulate things so we can experiment and verify hypotheses. Cognitive science has shown us over and over again that active participation, assigning meaning, and association to previously understood material is the best way to produce long-term learning and understanding.

WHAT IS IT ABOUT OBJECT-ORIENTED TECHNOLOGY THAT PRODUCES PRODUCTIVITY GAINS?

So what else is wrong with keeping our old approaches to analysis and design and switching to C++? Rebecca Wirfs-Brock[3] pointed out that the primary benefits of object-oriented programming have been our ability to reuse, refine, test, maintain, and extend the code. The ability to accomplish these has decreased as the size and, therefore, the complexity of applications has increased. It is the ability to manage this complexity that is the promise of object-oriented technology. However, if we first start by using functional decomposition we have defeated the benefits of objects before we have started. Using C++ to code many functionally decomposed units does little in the way of providing the benefits of abstraction. Wirfs-Brock points out that the main form of abstraction by which complexity is managed in object-oriented programming is encapsulation or information hiding.

Think about a complex human activity like playing the piano. At the lowest level of behavior, the fingers must be able to master complex patterns and movements. However, for the pianist to perform, he or she has represented this information at a much higher level. If the pianist had to consciously think about each note then the playing of the piece would be impossible. Hiding the implementation details of objects allows the programmer to think about the system at a much higher level and thus become capable of dealing with greater complexity. Extending the analogy to an organization, we can think about a chief executive and the hierarchies of responsibility and knowledge that exist within a company. If the CEO wants to start a new ad campaign, he or she should not need to possess all the knowledge of the graphic design artist who will create the ad. And when the CEO wants a change made to the ad, he or she should not need to be concerned with the implementation details that the graphic design artist must know to produce the new version of the ad.

Abstraction and encapsulation are complementary concepts, as pointed out by Grady Booch[4] in his book *Object-Oriented Design: Abstraction*. Booch focuses on how the outside view of an object and its encapsulation prevents clients from seeing its inside view in which the behavior of the abstraction is implemented. The problem with taking the approach of conducting a requirements analysis and functional decomposition is that the designer focuses on the details of implementation. It is these very details that we want to keep insulated from the designer so he or she may be able to deal with a complex system through abstractions. The CRC approach helps get people used to a new way of thinking by enforcing a system that by definition leads people away from implementation details and helps them move toward what Wirfs-Brock has compared to the client/server model. The cards focus on which responsibilities a given class has for its collaborators rather than on the details of how those responsibilities are carried out. Thus, encapsulation is enforced during the design phase. If implementation details were considered at this early stage then collaborators would have to know about those details, and that knowledge would have to be adjusted across related collaborators every time a change was made. This would be the opposite of encapsulation.

OOP IN A NON-OOP WORLD.

This is the title of an article by Steve Sheets[5] in which he describes some of the details about how he was able to use object-oriented concepts without the necessity of implementing in an object-oriented language. When Sheets joined his project, an alpha version of the software had already been written in Think Pascal—without object extensions. No object-oriented programming languages or environments were available at the time. On the whole, while Sheets' article is aimed at programmer readers who may want to see actual examples of inheritance, instancing of objects, and data encapsulation using some new data structures in a nonobject-oriented language, I found his story interesting because of his description of work he had done for America Online. *MacWeek* had just published a story about America Online and the Tribune, a Chicago-based media company. These companies had outmaneuvered the Baby Bells and delivered interactive online services with a local focus.

Remember the large telecommunications company I visited? They were adopting C++, but wanted to stay with their traditional approach to requirements analysis and functional decomposition. The group I visited had been chartered to develop software to supply exactly the kind of local-focus information services that America Online was already selling. It will probably be some time before the communications company gets their software ready—especially if they cling to their old analysis and design methods.

This can be taken as an object lesson. There is an additional hidden cost to taking the wrong approach to adopting object-oriented technology: missed opportunity when your competition beats you to the market. While there is no simple or painless way to adopt object technology, there is clearly something more significant than changing programming languages. To succeed in making the transition, we first have to change our analysis and design processes. We have to think object-oriented before we can become object-oriented. ≡≡

References

1. Beck, K. and W. A. Cunningham. Laboratory for teaching object-oriented thinking, *OOPSLA 89 Proceedings*, October 1989.
2. Gibson, E. Objects—born and bred, *BYTE*, 245–254, October 1990.
3. Wirfs-Brock, R. Object-oriented design: a responsibility-driven approach, *OOPSLA 89 Proceedings*, October 1989.
4. Booch, G. *Object-Oriented Design*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1991.
5. Sheets, S. OOP in a non-OOP World, *MacTutor*, 46–49, September, 1991.

...continued from page 8

In my opinion, Smalltalk should be a first programming language. It should be taught in elementary school just like Logo has been taught. As students progress, more classes should become available to them.

A few commercial realities might just stimulate the students who went to the library to read this newsletter. A trained O-O software engineer is worth 30% more than a C or Pascal programmer; the same holds true for new graduates. I know of several companies that have paid O-O designers and architects six-figure salaries.

CORPORATE DIRECTIONS

At the same time, the greatest disappointment must be the ongoing confusion created by IBM's "buy every horse in Kentucky" strategy. Once purchased, every horse seems to become instantly lame.

Colin Crook's keynote at Object World was the event that demonstrated the least leadership. Here is a man who agrees to provide the keynote address at the first major convocation of MIS folks interested in objects, and he fails to even recognize some significant use of objects within his own company, including a multimedia application development environment written in Smalltalk.

PREDICTIONS

Last year I made a few predictions that we should revisit in four more years. This year I think we will see a near continuous stream of corporate partnerships, mergers, acquisitions, and some companies going into the tank. This year is also the one that the Redmond boys get into the fray when Microsoft provides a C++ compiler for the first time.

I only have two predictions for the coming year. Sun Microsystems and its newly formed band of object-orienteers will develop the Self programming language into a viable competitor for C++. It holds some real hope of being a clean, efficient language for which tools can be built.

My other prediction is that in 1992 no company will ship a library of more than 100 classes where every method of every class, including all inherited methods, has been tested against a written spec.

Think of that as a challenge for our forming industry: help improve the quality of our products and the success rate of our users. ≡≡

INTERVIEW ≡≡

From classroom to shipped product in 20 months

by Robert Shelton

Momenta Corporation has brought to market one of the real product successes of 1991 (see "1991—The year it all changed," by Tom Love). Product development started April 1990, and Momenta shipped it in December 1991.

In this interview with Momenta Director of Software Development Beatriz Infante, several key success factors stand out. Momenta understood and stuck to its business strategy. Momenta knew that it is in the systems-solutions business, not the technology business. Momenta was determined to meet time-to-market objectives, and scratch-learned the technology that best fit its objectives: Smalltalk. Its pentop computer introduction foreshadows the savvy combination of business smarts and object technology that can give organizations a competitive edge.

—Editor's Note

HOTLINE: What exactly is your pentop product, and how does it differ from other recently introduced pen-based computers?

INFANTE: It's a 386SX running DOS that seamlessly supports a pen-centric interface. Unlike a number of other pen-based computing companies, we took a different approach in terms of our business objectives. Many others oriented themselves toward a tablet concept. We looked at combination of a penpad and laptop. We did this because of study group feedback. The problem we encountered was that if you placed a penpad in front of a person used to a personal computer, you'd get statements like, "It's an etch-a-sketch." When positioned as a penpad, the product was not viewed as a serious computer. When asked how much they'd pay for one, people placed it at a low price point. Position it as a 386SX with a keyboard, running DOS and Windows 3.0, with a complete pen-centric applications suite, and you get a very different value proposition back from the potential customer. The convenience of the pen shifts from being a toylike feature to a significant advantage in the customer's mind because it is now being added to the power of a full-featured laptop. People will ultimately want a commodity penpad, but the technology is not yet there to support that. Eventually, the notion of a very low-end product that is a penpad is very appealing, but the business aspects don't justify that product today.

HOTLINE: Who is your target pentop customer? Are they using the pentop as an electronic clipboard in away-from-the-desk industrial settings or in an office environment?

INFANTE: We're not addressing the blue-collar vertical market. That's a viable market, but we wanted to position this as a product for the horizontal market and the white collar worker. The pentop value proposition becomes very significant there, but a penpad just doesn't have the value. You add value by integrating well with existing PC technology, and add value with a pen-centric application suite that has an appliance-like simplicity to its user interface.

HOTLINE: What were your design objectives and where did object technology fit in?

INFANTE: Given the target market, there were certain givens about the unit. Full PC compatibility was required for the product to easily fit into an office running DOS and networking like Novell. DOS is not user friendly,

Robert Shelton is the editor of the Hotline on Object-Oriented Technology. He can be reached at (415) 928-7067.

Beatriz Infante is Director of Software Development at Momenta. She is one of the company founders and was instrumental in both software and hardware development. She may be reached at (415) 969-3876.

≡ Your Turn ≡

How about it? How are we doing? This is the place to send us your message. Tell us what you like and what you don't, what you'd like to see but don't — or whatever other feedback you'd like to send. Send comments to Robert Shelton, 1850 Union Street, Suite 1584, San Fransisco, CA 94123; Fax: (415)928-3036; or, if you're ordering, send along with your order.

Please check whichever box applies:

	Read entirely	Scanned	Found helpful	Will refer to	Didn't read
Feature	❑	❑	❑	❑	❑
Standards	❑	❑	❑	❑	❑
Interview	❑	❑	❑	❑	❑
Application Strategies	❑	❑	❑	❑	❑
Objects Down Under	❑	❑	❑	❑	❑
FYI	❑	❑	❑	❑	❑
Product Announcements	❑	❑	❑	❑	❑
Calendar	❑	❑	❑	❑	❑

Please rate the following **1** (dislike or strongly disagree) to **10** (like or strongly agree):

Presentation of material __ Comments: _____

Accuracy of material __ Comments: _____

Overall helpfulness in your job __ Comments: _____

What else would you like to see in THE HOTLINE? _____

Any other comments? _____

_____

_____

Name, title & phone (optional): _____

hotline on OBJECT-ORIENTED *technology*

❑ **Yes,** plug me into the latest thinking and developments in object-oriented technology. Enter me as a subscriber at the term marked below and rush me the current issue. This is a risk-free offer — I may cancel my subscription at any time and promptly receive a refund for the unused portion.

1 year (12 issues) ❑ $249

2 years (24 issues) ❑ $478 *(save $20)*

(outside US add $30 per year for air service)

Back issues @ $25 each *($27.50 foreign)*:
Vol.1, Nos. _____
Vol.2, Nos. _____ Vol.3, Nos. _____

❑ **Phone order**
Call Subscriber Services
(212)274-0640

❑ **Fax order**
Fill out term and name/address information, then fax to:
(212)274-0646

❑ **Send me a copy of** THE INTERNATIONAL OOP DIRECTORY @ $69 ($81 foreign)
(all directory orders must be prepaid; fax or mail credit card information to SIGS, 588 Broadway, #604, NYC 10012; make check payable to OOP Directory. Foreign orders must be prepaid in US dollars drawn on US bank.)

❑ **Bill me**
❑ **Check enclosed**
❑ **Credit card orders**
❑ MasterCard
❑ Visa

Mail (and make out check, if applicable) to:
THE HOTLINE
Subscriber Services
P.O. Box 3000, Dept. HOT
Denville, NJ 07834
(foreign orders must be prepaid in US dollars drawn on a US bank)

Card# Expiration Date

Signature

Name

Company

Street/Building#

City/Province

ST/Zip/Country

Telephone

Send me a complimentary copy of your related publication(s):

❑ THE JOURNAL OF OBJECT-ORIENTED PROGRAMMING

❑ OBJECT MAGAZINE

❑ THE C++ REPORT

Call Subscriber Services
(212)274-0640
or send order to
SIGS, 588 Broadway, #604
New York, NY 10012

D2CC

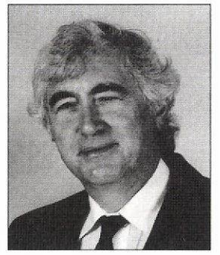
hotline on OBJECT-ORIENTED *technology*

VOLUME 3, NUMBER 5 THE MANAGER'S SOURCE FOR TRENDS, ISSUES & STRATEGIES MARCH 1992

A large-scale users' assessment of object orientation

The Object Interest Group (OIG), consisting of 14 large U.K. companies, two Government departments, and one university, was formed in May 1990.

Its purpose was to understand and assess the technology, get hard evidence to support the assessment, enable member companies to position the technology in their IT strategy, and tell suppliers what large-scale users want in terms of method, tools, and support.



Norman Plant

Project members were Barclays Bank, British Petroleum Research, British Aerospace, British Airways, British Steel, British Telecom, Department of Trade and Industry, The Treasury, Ford Motor Company, Imperial Chemical Industries, Lloyds Bank, National Westminster Bank, Prudential Assurance, Rolls Royce, Royal Insurance, Trustee Savings Bank, and University of Strathclyde. No funds were involved. Each company agreed to contribute at least 40 man-days; in total, some 1500 man- days went into the assessment. The business case for sharing was at least 40 days in, 1500 out.

The subjects researched in this study were:

- analysis and business modelling
- methodologies and CASE
- design
- languages
- development support
- hardware and software platforms
- object libraries
- database technology
- distribution
- standards
- project management
- IT development organisation
- culture change and training

The nature of the research was to apply where relevant the following questions to each of the above subjects:

- What is it?

continued on page 4 ...

IN THIS ISSUE

FEATURE — A large-scale users' assessment of object orientation *Norman Plant*	1	2	FROM THE EDITOR
STANDARDS — Report on the Object-Oriented COBOL Task Group *Megan Adams*	7	10	Discardable software systems in the object age *Frank Douglas*
INTERVIEW — K.C. BRANSCOMB *interviewed by Robert Shelton*	13	19	OBJECTS DOWN UNDER— The Australian Object-Oriented Scene *B. Henderson-Sellers*
BOOK REVIEW — OBJECT-ORIENTED SYSTEMS ANALYSIS, MODELING THE WORLD IN DATA *reviewed by Michael Fuller*	20	22	PRODUCT ANNOUNCEMENTS
FYI — Industry publication excerpts	25	27	CALENDAR

From the Editor



For vendors of object technology, 1991 was the first year of truly explosive growth in general interest and demand. The phrase incessantly heard was "I can't believe how many people...." Raw numbers, though, masked an even more interesting phenomenon: many of those newly exploring the technology were from organizations previously thought too conservative, traditional, COBOLized, and Yourdonesque to take object technology as anything other than a long-hair California phenomenon.

The upward trend for object technology is accelerating, and 1992 is already shaping up as OOT's year of commercialization. Object technology is breaching the mainstream in some unexpected ways:

- Intellicorp and James Martin & Company announced in February their collaboration to blend Mr. Martin's new concepts for object-oriented planning, analysis, and enterprise engineering with Intellicorp's PROKAPPA, an object-oriented visual development environment that generates application code in standard ANSI C.
- OO COBOL promises this year to bring object technology to the heart of MIS: the COBOL development teams responsible for maintaining the legacy systems that represent the single largest body of application code that has been written in any language.
- Allen-Bradley Company, one of the largest manufacturers of process control systems, has been working with ParcPlace Systems ObjectWorks Smalltalk, primarily to cope with software complexity and time-to-market demands.

Major established corporations like British Airways, Barclays Bank, Ford Motor Company, Rolls Royce, and British Telecom are discovering that object technology is both powerful and dangerous—success belongs to those who know how to manage this new power tool.

In this month's feature, **Mr. Norman Plant,** Chairman of the Object Interest Group and formerly of British Airways, reports on a substantial joint study by 14 large UK companies and the British Government. Their conclusion: object technology must be reckoned with, but it also must be managed effectively. The caliber of management and discipline required to succeed with object technology exceeds that required to successfully apply technologies like information engineering. In this issue, Mr. Plant discusses the primary conclusions and direction of the study. In future issues, he will join us again to share more of the detail and recommendations. The scope and level of this study should give notice to organizations just beginning to look at object technology: absorbing this technology into your strategic tool kit is a necessary and profitable challenge, but don't underestimate the challenge.

In last month's feature article, Mr. Donal O'Shea articulated one of the key cultural success factors in meeting this challenge: at companies that have successfully integrated object technology, the developers have offered object technology solutions instead of high-technology separatism. In many companies, OT has entered the mainstream in part because it's practitioners have not allowed their technology to become a point of alienation from their colleagues. Also, their solutions are able to address complex business demands that are viewed as otherwise technically intractable. In short, the successful have been those who have made an impact as *team members*. Cultural integration is a theme echoed by several authors in this month's *Hotline*, and is particularly critical, as Mr. Plant hints, in organizations that are building a shared corporate information base.

Editor
Robert Shelton

SIGS Publications, Inc. Advisory Board
Tom Atwood, *Object Design*
Grady Booch, *Rational*
George Bosworth, *Digitalk*
Brad Cox, *Information Age Consulting*
Chuck Duff, *The Whitewater Group*
Adele Goldberg, *ParcPlace Systems*
R. Jordan Kreindler, *General Electric*
Meilir Page-Jones, *Wayland Systems*
Tom Love, *Consultant*
Bertrand Meyer, *Interactive Software Engineering*
Sesha Pratap, *CenterLine Software*
P. Michael Seashols, *Versant Object Technology*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

Hotline Editorial Board
Jim Anderson, *Digitalk, Inc.*
Larry Constantine, *Consultant*
Mary E.S. Loomis, *Versant Object Technology*
Reed Phillips, *Knowledge Systems Corp.*
Trygve Reenskaug, *Taskon A/S*
Zack Urlocker, *Borland International*
Steven Weiss, *Wayland Systems*

SIGS Publications, Inc.
Richard P. Friedman, Founder & Group Publisher

Art/Production
Kristina Joukhadar, Managing Editor
Susan Culligan, Creative Director
Elizabeth A Upp, Production Editor
Caren Polner, Desktop Designer

Circulation
Diane Badway, Circulation Business Manager
Kathleen Canning, Fulfillment Manager
Vicki Monck, Circulation Assistant
John Schreiber, Circulation Assistant

Administration
David Chatterpaul, Bookkeeper
Frank Ferrantelli, Bookkeeper
Suzanne W. Dinnerstein, Conference Manager
Laura Taylor, Conference Coordinator
Jennifer Fischer, Public Relations
Jennifer Englander, Administrative Assistant

Margherita R. Monck, General Manager

Jane M. Grau, Contributing Editor

hotline on OBJECT-ORIENTED *technology*

The Hotline on Object-Oriented Technology (ISSN #1044-4319) is published monthly by SIGS Publications, Inc., 588 Broadway, NY, NY 10012, (212)274-0640. © Copyright 1992 SIGS Publications, Inc. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the U.S. Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher. Mailed First Class. Subscription rate — one year (12 issues) $249, Foreign and Canada $279. Single copy $25.

POSTMASTER: Send address changes & subscription orders to The Hotline, Subscriber Services, P.O. Box 3000, Dept HOT, Denville, NJ 07834.

Submit editorial correspondence to Robert Shelton, 1850 Union St., Ste. 1548, San Francisco, CA 94123. Voice: (415) 928-5842; Fax: (415) 928-3036.

SIGS PUBLICATIONS

Publishers of *Hotline on Object-Oriented Technology, Journal of Object-Oriented Programming, Object Magazine, The X Journal, The C++ Report, The Smalltalk Report,* and *The International OOP Directory.*

Back issues available

Vol..3, No.4/February '92 ≡ The big prize: acceptance of O-O by the MIS community • *O'Shea* ≡ Retrospective: 1991—The year it all changed • *Tom Love* ≡ Making the transition to O-O technology • Ron Suarez, PhD ≡ Interview with Beatriz Infante • *Shelton*

Vol. 3, No. 3/January '92 ≡ Enterprise object modeling: knowing what we know • *Shelton* ≡ Adopting objects: pitfalls • *Connel* ≡ Adoption rate of object technology: a survey of NSW industry • *Henderson-Sellers.*

Vol..3, No. 2/December '91 ≡ Accepting object Technology • *Bennett* ≡ Adopting objects: a path • *Connel* ≡ Incorporating graphical content into multimedia presentations • *Berman & Weiss*

Vol.3, No. 1/November '91 ≡ Leading the U.S. semiconductor manufacturing industry toward an object-oriented technology standard • *Hollowell* ≡ Coping with complexity: OOPS and the economists' critique of central planning • *Leavoie, Baetjer, & Tulloh* ≡ Choosing Object Technology: What's the object? • Rosenthal ≡ OOP: the MISsing link • *Rovira*

Vol.2, No. 12/October '91 ≡ A modest survey of OOD approaches • *Bulman* ≡ What is a "certified" object programmer? • *Bellin* ≡ Perspective: investing in objects today •Bowles ≡ Object oriented in Melbourne, Australia • *Haebich* ≡ The Object Management Group • *Gutman & Mathews*

Vol.2, No. 11/September '91 ≡ From applications to frameworks • *Urlocker* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Getting started with object technology: efffectively planning for change • *Lorenz* ≡ Object statistics on the way • *Lenzi* ≡ On objects and bullets • *Onart*

Vol.2, No. 10/August '91 ≡ Distributed object management: improving worker productivity • *Osher* ≡ Getting the best from objects: the experience of HP • *Coleman & Hayes* ≡ Applications: EC employs object technology ... • *Stein* ≡ Capacity Planning: Fiddling while ROMs burn • *Rovira*

Vol.2, No. 9/July '91 ≡ Multimedia is everywhere! • *Weiss & Berman* ≡ Developing an object technology prototype • *Kulinek* ≡ Object-oriented capacity planning • *Rovira* ≡ How OOP has changed our developmental lifecycle • *Zeik* ≡ Modularization of the computer system • *Angenius*

Vol.2, No. 8/June '91 ≡ Domain of objects: the Object Request Broker • *Dyson* ≡ Object-based approach to user documentation • *Reeder* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Do we need object-oriented design metrics? • *Hopkins*

Vol.2, No.7/May '91 ≡ Hybrid object-oriented/functional decomposition for software engineering • *Henderson-Sellers* ≡ So, what makes object databases different? (Part 4) • *Blakey* ≡ Using the generic application to solve similar domain problems • *Gossain* ≡ Experiences using CLOS • *Hopkins* ≡ International Conference on Object-Oriented Technology, Singapore • *Bennett*

Vol.2, No.6/Apr. '91 ≡ An artist's perspective of programming • *Rovira* ≡ So, what makes object databases different? (Part 3) • *Blakey* ≡ Moving from Pascal to C++, Part 3 • *Gole* ≡ Object projects: what can go wrong • *Stewart* ≡ Reflections from LOOK-'91 • *Anders-Angenius*

Vol.2, No.5/Mar. '91 ≡ Large-scale object systems development • *Morgan & Davis* ≡ Data definition languages • *Blakey* ≡ Object disorientation • *Kerr* ≡ Moving from Pascal to C++, Part 2 • *Keller* ≡ The Object Management Group • *Barber*

Vol.2, No.4/Feb. '91 ≡ Moving from Pascal to C++ • *Sollisch* ≡ The Australian object-oriented scene • *Henderson Sellers* ≡ So, what makes object databases different? • *Blakey* ≡ Managing the evolution • *Nowobilski* ≡ 1990: the year of realism • *Love* ≡ Choosing a configuration for object development • *Moody*

Vol.2, No.3/Jan. '91 Special issue: CASE and Methodology ≡ Object orientation: stop, look and listen! • *Page-Jones* ≡ The spiral model for object software development • *Wasserman & Pircher* ≡ Bootstrapping object-oriented CASE • *Wybolt* ≡ A CASE tool for object software development • *Ackroyd* ≡ Comments on "Race for the database pennant" • *Loomis*

Vol.2, No.2/Dec. '90 ≡ Waiting for OODB2 • *Durham* ≡ Object modeling • *Singh* ≡ Object-oriented COBOL — the next generation • *Adams & Lenkov* ≡ Object-oriented training in the UK • *Hopkins*

Vol.2, No.1/Nov. '90 ≡ Objects by teamwork • *Constantine* ≡ Race for the database pennant: a third-generation manifesto • *Stein* ≡ Project team descriptions *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.12/Oct. '90 ≡ Crown jewels: the value of OOT in MIS departments • *Bonar* ≡ Why is an object database faster than a relational system? • *Loomis* ≡ Visual programming object language environments • *Klausner* ≡ Hype — necessarily (or a necessary) evil? • *Adams* ≡ On Managing: Déjà Vu • *Lenzi*

Vol.1, No.11/Sep. '90 ≡ Introducing OOPS through design — it's all in the words • *Copen Goldstein* ≡ The Australian object-oriented scene • *Henderson-Sellers* ≡ Objects on tour (Review of Borland's OOP World Tour) • *Lenzi* ≡ Design principles for modular product development • *Bennett* ≡ Asset management and object-oriented technology • *Hopkins & Warboys*

Vol.1, No.10/Aug. '90 ≡ WyCASH+: An application built within an OOP environment • *McCullough & Deshler* ≡ Mathematics and methodology • *Martin* ≡ Organizing for reuse • *Goldberg & Rubin* ≡ CALS Expo–89 Conference • *Purdy*

Vol.1, No.9/July '90 ≡ Cognos and Eiffel: a cautionary tale • *Leathers* ≡ What's in an object anyway? • *Novobilski* ≡ The impact of Windows 3.0 on the OOP market • *Urlocker* ≡ Introducing reuse into the development process • *Goldberg & Rubin*

To order, see back page; for reprints of individual articles, call Michael Biggerstaff at Reprint Management Services (717) 560-2001.

Calendar

The Hotline Calendar presents conferences and meetings that focus exclusively on object-oriented technology. To have a meeting or conference listed, please send the dates, conference name and location, sponsor(s) and contact name and telephone number to the Editor: Robert Shelton, 1850 Union St., Ste. 1548, San Francisco, CA 94123, Fax (415) 928-3036.

March 11–18, 1992 **CEBIT** Hannover, Germany Sponsor: Hannover Fairs USA, Inc. Contact: (619) 987-1202	**March 15–20, 1992** **ANSI C++ Committee Meeting** London, UK Contact: (408) 447-5279	**March 24–26, 1992** **DB/Expo '92** San Francisco, CA Sponsor: NDN Contact: (415) 941-8440	**March 31–April 4** **Patricia Seybold's Annual Tech Forum** Cambridge, MA Sponsor: Patricia Seybold's Computing Group Contact: (617) 742-5200	**April 27–30** **XWorld** New York, NY Sponsor: SIGS Publications, Inc. Contact: (212) 274-0640

Multimedia

...[Peter] Meng [co-founder of Drochelman/Meng & Associates, a multimedia integration company based in St. Louis] commented on a recent TIME Magazine article about multimedia which expressed the opinion that what multimedia lacks is the so-called "killer application"—something like the electronic spreadsheets or word processors that triggered the personal computer revolution. Meng thinks that this "killer application" will turn out to be not an application at all but an object-oriented operating system that gives users the capability of creating their own customized applications. One example he gives is that a user could create an individual application such as a spreadsheet with the numbers animated, along with sound accompaniment. Meng predicts that the recently announced alliance of Apple and IBM will do much to enhance and move forward the whole field of interactive multimedia, particularly in the matter of standards and system compatibility. Potential uses of interactive multimedia are virtually limitless, especially in the field of education and corporate training. Meng says, "In our fast-moving society, where it's vital to educate and re-educate people quickly, multi-media offers itself as an incredibly powerful training tool."...As for the future of multimedia, [co-founder Peter] Drochelman says that even though we still live in basically a print world, he thinks that's changing. "I see multimedia replacing print. Television has been an interim step in this process. Eventually, I see the computer becoming an all-encompassing technology, integrated in tho totality of our daily lives."

Drochelman/Meng & Associates: a media resource integration firm, Carol Ellerman, ***St. Louis Computing,*** *12/91*

Languages

Outside of military contracts, where it's mandated, the Ada programming language has had little impact in the real-time embedded market. The Ada 9X revision effort, which reaches a new stage next month, may help make the programming language a much stronger contender in that arena...Resistance to Ada as an embedded programming language runs deep. "It's big, it takes up a lot of memory, and the run time is relatively large," said Cadre's [Wes] Hair. "If things need to happen at a 15-microsecond interval, the rendezvous [Ada task-synchronization mechanism] is out of the question." For these reasons, he noted, many embedded designers interested in an object-oriented approach are leaning toward C++. But Ada also has its hard-core supporters, Hair noted. "Some people feel the type checking and error handling of the compilers provide a tighter piece of code that's easier to maintain and support. You don't have as many testing or debug problems."...Intermetrics' [Tucker] Taft argues that Ada provides significant advantages over C and C++ for embedded designers. "Ada will provide more programmable support for multiprocessing and multitasking applications," he said. "It's also safer. With C and C++, you're still fiddling with pointers and you can easily point to random junk. Debug can be a nightmare."....

Ada 9X could get a bigger real-time role, ***Electronic Engineering Times,*** *12/2/91*

Interoperability

..."Portability is crucial," says Ron Suarez, president of Arbor Intelligence Systems Inc., Ann Arbor, Mich., "especially for corporations that have already bought into different hardware platforms. Where a group of developers may be sitting at Unix workstations and people in accounting are using PCs and the graphics department is using Macs, you may have certain applications that they should all be able to use." That portability brings important benefits to VARs and developers building products with OOP as well. "For a small company like us, working with these kinds of tools means that we can reach a much wider market," points out Suarez. "We can sit at our machine of choice, which is the Mac, and develop applications that will run on Unix machines and on Microsoft Windows."..."Our product, Smalltalk Nexpert Bridge, allows one to imbed a knowledge base produced using Nexpert within an application created with Smalltalk," says Suarez. "The programs that you create are instantly portable from the Mac operating system to Microsoft Windows to Unix workstations So a knowledge base developed with Nexpert can be used across all those platforms."....

Brave new VARS, Jim Waddell, ***VAR Business,*** *12/91*

Antihype

...As a result of controllers and views, the application object is much less in control of its own destiny. It will only handle a subset of the events that it otherwise might. When it does respond to an event, the response is likely to be restricted to the application data. The days when the application ruled the interface are over. Losing autonomy isn't bad in itself. Given the enormous complexity of GUIs, overall control is best left to the application-independent GUI classes. By reusing view and controller classes, designers and programmers can largely limit their attentions to the design of application classes. However, reuse means that a programmer no longer has the control that comes from creating a program from scratch. As major league control freaks, programmers may find this pill a bit hard to swallow...

...The secrecy and lack of attention associated with GUI issues is ironic in the extreme. OO was largely created as a reaction to command line-based—DOS, for example—user-computer interfaces. OO pioneers like Alan Day wanted user interfaces that provided direct manipulation capability. This means that the objects the user manipulates in the "real world" are used in the computer system's interface. (Flight simulators are good examples of this concept. Virtual reality is the ultimate application of direct manipulation.) Object orientation is more than the communicating object the programmer sees. It is a whole new way for most users to interact with computer systems. Dynamic binding is a mind bender. But, there is also a significant paradigm shift when command line or character-based hierarchical interfaces are converted to GUIs. Unlike dynamic binding, however, the GUI part of the paradigm shift affects everyone—user, software developers, and methods specialists. The universal impact of the shift to GUIs is the greatest dirty little secret. No one will be spared the conceptual devastation of the OO paradigm shift, not even the users. How long before we hear the sound of shattered glass because users are trying to click their mice on their windows?

Antihype: OO's dirty little GUI secrets, John Palmer, ***Object Magazine,*** *1-2/92*

Whether or not one agrees with Mr. Martin's approach to shared information through Information Engineering (or the writings of his graduate student assistants, as the case may be...), Intellicorp's dynamic CEO, **K.C. Branscomb**, is clearly on target in observing that Mr. Martin lends significant credibility to object technology with the MIS customer. Intellicorp's strategy is of particular interest because they are integrating with MIS from two directions: information engineering from the top, and standard ANSI C from the bottom. Change by integration and evolution instead of rejection and revolution—this approach is remarkably similar to that of the relational database (RDBMS) vendors, and accounts for much of their success today. Ms. Branscomb underscores this theme of integration into the existing technical, social, and application environments in our interview with her in this issue. It's a concept as critical to Intellicorp's future success as it is to the success of object technology in general.

Ms. Megan Adams of Hewlett-Packard's California Language Laboratory brings us a report on a different approach to making object technology viable for MIS. With a standard being established for O-O COBOL, MIS organizations will have yet another avenue by which to absorb object concepts into their existing technology and application base. As the first standards-based O-O COBOL products become available over the upcoming months, prospective users will likely be asking:

- Will the language definition avoid the problems experienced with C++? To what degree can the developer bypass object concepts and violate the data structures?
- Will O-O COBOL give a new lease on life to CASE produces like KnowledgWare's ADW and Texas Instruments IEF? Will such products become "objectified" by generating O-O COBOL or C++ instead of ANSI COBOL or C—and therefore miss the point of object technology altogether?
- Will the predominant use of the new language products be for program-level reengineering or new application development?

Ms. Adams article gives us an indication of what the new language standard will provide, but for answers to questions like these, we'll follow the products as they become available.

O-O COBOL is another entrant in the field of hybrid languages that blend object-oriented constructs into a traditional procedural language syntax. PROKAPPA is a hybrid of a different stripe: the user applies object-oriented thinking, but the application programs are generated in a pure procedural language. What I find most intriguing is the resurgence being experienced by the pure object environments. While not boasting integration or direct compatibility with existing application source code, the best pure environments sport a capability that only visual tools like PROKAPPA can hope to match: raw development speed. They also offer a powerful constraint that hybrid systems cannot: the object paradigm cannot be bypassed. This is a blessing or a curse, depending on your point of view. Ms. Branscomb builds a strong case challenging the viability of object purity when one's objective is selling to MIS. From the viewpoint of organizations like those discussed in Mr. Plant's feature, having three distinct entry points into the technology significantly enhances it's approachability.

Ah, the subject of language! For many developers, Smalltalk is as different from ANSI C as C itself must have seemed from COBOL when MIS developers first used it on some "experimental" applications; but companies like Allen-Bradley and Momenta Computers [Ed: see *HOTLINE*, February 1992] have determined that time-to-market is more critical than tradition. Both organizations represent a trend by established C-language developers toward Smalltalk instead of C++. To me, the underlying trend is away from language-based development altogether. Visual or model-driven specification techniques and extensive, specialized class libraries will be the "languages" of the next generation of commercial developers. Most application developers will use raw language only for "glue" code—integrating purchased or generated application components; and, perhaps, for hand writing a few, highly specialized classes or methods.

In his article on discardable applications, **Mr. Frank Douglas** carries this theme. "The rules for doing business are changing radically," he says, and "The rules for building applications will also change... If developers of business applications cannot adapt to the new, more volatile business environment, they, themselves, will be discarded." Mr. Douglas points to the trend away from traditional language development toward environments like Smalltalk and Whitewater's Actor. He also challenges some of the fundamental assumptions most of us hold on application lifecycles, raising the specter that, while Mr. Martin may bring credibility to the information engineering-based lifecycle for the MIS organization of today, a very different reality may govern the MIS organization of the not-so-distant future.

Cultural integration is first of four trends that I expect will shape object technology during 1992. The others are cooperative alliances, reduced emphasis on language-based development, and increasingly tight time-to-market requirements. In this issue, our authors address all four of these. I think Mr. Douglas gave us a healthy perspective in his observation on business today: the rules are changing, and so will the rules for systems development. We ourselves can choose to change or fall behind. Business, as usual, will be pleasantly Darwinian! ≡≡



...continued from page 1

- Why is it different?
- Why use it?
- What are the benefits and costs?
- What should it be applied to?
- Why now?
- Who has used it?
- What were their experiences?
- Who supplies supporting products?
- How do they compare?
- Who is pushing it forward?
- How do you get to it from where we are today?
- What is preventing faster exploitation?

Some 35 practitioners and major suppliers worldwide were consulted for hard evidence. The draft report was issued for internal distribution in March 1991; an edited version is to be issued to participating suppliers and practitioners during February 1992.

The report was substantial because it covered so many subject areas at a detailed technical and business level. In this article I shall focus on our overall conclusion relating to how object orientation addresses the current problems of a large IT division (of course, new ones will appear), how it can more effectively reflect the business of the enterprise, and the business opportunity it enables. Later articles can focus on important topics such as "what we want from suppliers."

OUR CONCLUSION IN BRIEF

I shall extract one quotation that summed up our shared view from the report:

> Object orientation is potentially one of the most powerful technologies ever to become available to the IT Industry and its users. As such it demands high calibre management. It is not a panacea but a high power tool—dangerous if misused but capable of great things.

The remainder of the article will explain and develop this conclusion. It is presented from the point of view of large IT divisions (organisations comprised of more than 500 development staff).

IT DIVISION PROBLEMS AND THE POTENTIAL IMPACT OF OBJECT ORIENTATION

The following problems face all members of the OIG:

- Despite the tendency for IT spending to increase each year as a proportion of corporate revenues, IT divisions are still not perceived as responsive to the changing demands of the business.
- Maintenance of the existing applications portfolio leaves only a fraction of the development staff available for new application work.
- The classic "waterfall" lifecycle means that projects have to be committed to a specification well in advance of delivery, take a long time to come to fruition, and may well have become inappropriate by the time they are delivered.
- Having to commit to solutions well in advance of their delivery makes it difficult to exploit opportunities offered by new technologies.

Some of the major causes follow below, together with how we concluded (with evidence) that object orientation will help.

Maintaining the portfolio

A typical application among our members will have absorbed 400 man-years of development effort over 20 years and a hundred or more people will have worked on it at some time. When a new (often young) person joins the team, it will be a year before he can be "let loose" with any confidence. Typically he will move to another project after 2 years.

A major part of the problem is the amount he has to know about the rest of the application before he can work effectively on the bit in front of him. We do not have an effective unit of modularity. The learning curve will have cost us $75,000 for one year of efficient support. The problem occurs because with our function and data model of the enterprise, part of the model (the data) lies outside our current unit of modularity (the program). With object orientation, the whole of the model lies within the new unit of modularity (the object) with well-defined interfaces independent of the application requirement. objects thus provide "fire walls," which prevent the effects of change spreading uncontrollably through an application.

Analysis Paralysis

A single part of a business domain in a large industry can be so wide ranging and complex that the analysis can go on for years, during which time the company will significantly reorganise its operations within the domain. "Analysis Paralysis" is a common ailment in most large IT divisions. object orientation makes a strong conceptual separation between domain modelling (creating the foundation model of the domain's fundamental stable concepts, e.g., customer, product, contract) and application modelling (meeting specific information needs). As the foundation model grows, applications will pick up more and more ready-modelled, enterprise- wide con-

FYI

Excerpts from leading industry publications on aspects of object technology

Databases

..."Object-oriented programming (OOP) and ODBMS are among the most important leading edge technologies today," says [Patrick] Arnone [president of PRC Open Technology Inc.]. "It's still a very small part of the industry, but we believe the market for OOP and ODBMS will grow throughout this decade. Right now we have to educate the community on the benefits of ODBMS, sort of like how the relational DBMS vendors did in the late '70s. It is not going to replace RDBMS technology, however, but rather will complement it very nicely." Advantages of object-oriented over relational database systems include better integration of text, sound, images and graphics, as well as features suited for highly interactive application. As such, ODBMS is expected to be at the core of tomorrow's multimedia and image processing software. "The early adopters, in both federal and commercial markets, will be in the engineering and scientific areas for computer-aided design and computer-aided engineering, certain segments of the MIS market, office automation system, multi-media network management, CASE, financial modeling and compound document imaging," Arnone offers as examples....

Brave new VARS, Jim Waddell, ***VAR Business****, 12/91*

Applications

...Any menu or window on the desktop can be hooked into and subclassed. This means that anything is fair game, whether it's the listbox in your application that you need to enhance, or the menu bar in Aldus Pagemaker that needs an extra command or two. I have to admit, at first this made me think a little about the legal ramifications. However, as long as you've actually purchased the application, nobody can really complain of any wrongdoing. After all, you haven't actually modified anybody's code —just the way it interacts with Windows and other objects...

Subclassing Applications, Mike Klein, ***Dr. Dobb's Journal****, 12/91*

Analysis & Design

[Lou Mazzucchelli of Cadre]:...Traditionally, people have tended to concentrate on the functional behavior or the process behavior. I mean, what happens when this data turns into this data and then it goes over here and gets turned into something else. What object-oriented analysis does is borrow a lot of ideas from database design and information engineering. Before you do that, let's understand what the data space looks like. What are the things that the system is manipulating? Why do those things exist? How do those things relate to each other? It turns out that if you do a good job of that, you're actually beginning to identify candidates for objects that you might design into a system to support this space. So there's been a lot of work in the last year or two on ways to represent the design of an object-oriented program and then ways to transform that design into a reasonable implementation: and you're beginning to see a few products that do things like that—you'll see more....

Q & A: Lou Mazzucchelli, Cadre's vice president, Gregg Wendorf, ***The Sun Observer****, 12/91*

In the Industry

..."Thirty percent of federal agencies are already using some kind of object-oriented programming (OOP) software," says Robin Rather, of Vienna, Va.-based Information Strategies Group, who recently completed a study of OOP in the federal government. She also discovered that another 25 percent of the agencies said they would be buying OOP technology in the next three years....

Company courts Federal market with object-oriented programming, ***Washington Technology****, 12/19/91*

...But as we head into the 1990s, there is a new category of computer specialist that is about to emerge, people I call paraprogrammers. This new type of programmer could create some interesting new job opportunities for a lot of people who do not have specific computer backgrounds, but are willing to learn some new skills in order to get into the world of computers and business. Other business fields have specialists like paralegals and paramedics, and now the computer industry is about to get its own: paraprogrammers. The role of a paraprogrammer is linked to a significant new software programming concept called object-oriented programming (OOP). With this approach to programming, much of the programming code is actually prewritten by the programming language vendor, and the person working with this code draws on the objects, or object libraries, to write the bulk of the specific program....We actually have a type of paraprogrammer today. These individuals normally reside within an MIS department and are given the task of customizing a database or spreadsheet... Many times they create macros for repetitive tasks. Some of these are actually professionally trained programmers, while many of them have just learned a particular database language and are trained to create this custom layer of the program...Trained programmers will continue to play a very important role, especially in the creation of code that will be used for the commercial market. They will also become an important part of the management of these paraprogrammers, in the same way doctors or lawyers oversee the work of their paraprofessionals...

Industry Insight: new category of computer specialist, Tim Bajarin, ***Computer Currents: San Francisco Bay Area,*** *12/3/91*

Partnerings & Acquisitions

NCR Corporation announced that joint development has been underway with **Lucid, Inc.** for future software application development environment products for the NCR System 3000 family of Intel-based, general-purpose computers. The Lucid technology is based on a client/server architecture and provides an integrated environment for developing and understanding C and C++ programs. NCR's announcement coincides with Lucid's announcement of its product program using this technology, the Energize Programming System. Energize includes browser tools developed jointly by NCR and Lucid over the past year.

Servio Corporation has retained **Booz-Allen & Hamilton,** the international management and technology consulting firm, to provide training and consulting services for users of Servio's object- oriented software products. Booz-Allen will work with Servio to enhance technical training programs, materials, and documentation for Servio's GemStone object database management system and GeODE application development products. In addition, Booz-Allen will provide technical instruction and consulting services for Servio's major clients.

Rational and **O_2** announced cooperative marketing agreements under which the companies will jointly market their products to object-oriented software development organizations. They will explore opportunities to produce a complete integration of Rational Rose, Rational's graphical object-oriented analysis and design tool, and O_2 OODBMS.

GUIdance Technologies announced it is expanding its European business operations through an exclusive distribution arrangement. The newly formed entity, **GUIdance Europe,** will market the Choreographer line of graphic user interface (GUI) development tools throughout Europe. Headquartered in Holland, GUIdance Europe is an affiliate of GUIdance Technologies, Inc. and **Software Generation International BV**, a UK-based organization. In connection with the arrangement, an equity investment has been made in GUIdance Technologies. Software Generation has successfully distributed US-based products in Europe, including those from ViaSoft, Intersolv, and Aion.

Sun Microsystems, Inc. and **Cray Research, Inc**. announced that the firms will cooperate in both software and hardware areas that will include sharing information and technology. The firms plan to create a seamless software environment that will allow Sun systems and Cray Research supercomputers to work together efficiently. Cray Research will use technology developed at Sun Microsystems to build high-end SPARC1 RISC computers that will complement the product lines of both companies. Cray Research has also joined SPARC International, an organization of more that 200 SPARC vendors that supports binary compatibility among products through the SPARC Compliance Definition.

The Object Management Group (OMG) has issued a Request For Information (RFI), to solicit documentation of object-oriented technologies in the area of Object Services. It was also announced that the specification for the Common Object Request Broker Architecture (CORBA) is now available to all interested parties at a cost of $50 through the OMG's Framingham, MA headquarters.

Hyperdesk announced agreements with several independent software vendors that plan to build and integrate open distributed applications with the HyperDesk Distributed Object Management System (HD-DOMS). **IXI Limited, Constellation Software, Inc.,** and **Uniplex Integration Systems** plan to incorporate the HyperDesk DOMs into future releases of their software products serving the end user PC, workstation, and server marketplace. HyperDesk and Netwise Corporation have also agreed to continue to work together to use Remote Procedure Call (RPC) technology in the HyperDesk DOMS. IXI and Uniplex are licensing the HyperDesk DOMS through HyperDesk's Premier Partners Program. Constellation is licensing HC-DOMS through a development and licensing agreement that gives HyperDesk the opportunity to relicense HyperStar.

Clarity Software of Mountain View, CA and **Mobius Computer Corp.** announced a marketing agreement to bundle Clarity's Rapport software with Mobius' Mirage Series of SPARC-compatible UNIX workstations. Rapport is an object-oriented program that combines work processing, spreadsheet, drawing, slide creation, audio, multimedia e-mail, and other features in a single package. The Mirage IPS system is based on the 25 MHzLSI SPARC chipset.

cepts. Applications in the same domain should integrate naturally because they are built on the same foundation of components. Thus object orientation lends itself to the enterprise's integrated applications architecture. The need for more systems integration is driving the redevelopment of many applications today and consuming much analyst effort with the intellectual difficulties of understanding the ever widening domain with our current "applications only" -based approach. The natural layering between the domain and the application, the natural modularity of object domain concepts, and the growing foundation of reusable designs means the object approach is evolutionary and less prone to paralysis.

It is true that other current approaches like information engineering make a strong conceptual separation between domain and application modelling. The difference with object orientation is that, despite the conceptual split, there is no disjuction between domain and application models. For example, with object orientation a component such as "profit" may exist in the domain model and then be directly reused in a number of different applications. With information engineering, "profit" would be unlikely to appear in the domain model although it may be required in an application.

Corporate Data

Most large IT divisions have objectives to establish corporate databases that avoid overlap and that users can exploit to gain good business insights. Plans and managers focussed on these objectives seem to come and go with relatively little impact. One reason has do with the application-based project team culture of the large IT division. Project managers are rewarded for delivery. They are not paid to slow their project down while another project catches up and is integrated around a more effective data architecture. Object orientation requires that some of the development teams are focussed around the production of business domain objects—as distinct from applications—that will lead to a more effective architecture. Users find difficulty today in exploiting data as effectively as they might, because the data they see on our current databases is divorced from its context and is meaningless in itself. Intermediate specialists between the true user and development evolve to handle the query, which is expensive. Through encapsulation of data and process, object orientation adds back the context dimension.

Quality

All large IT divisions have quality campaigns. However quality is defined, "fitness for purpose" has to be an essential component of its definition. Our current waterfall production cycle means that projects are committed to a specification well in advance of delivery. Two years is not uncommon for a large project. This is incompatible with "fitness for purpose" because the business is evolving continuously. IT managers that have posters on the wall about "delivering IT products and services on time, right the first time, etc." are kidding themselves. Object orientation, with its reuse of tried and tested components, shorter interactive development cycles, and more effective response to change, offers the hope of properly implementing total quality management. A good analogy is that it enables software production to mature to a state analogous to the production line in engineering manufacturing. The "cost of quality" is known to be around 20% in manufacturing and currently around 60% in software production. The potential for IT unit-cost saving is substantial.

Computing "horse power" from hardware

Many chief executives are sceptical over the value of budgeting money for intelligent workstations for their business users. Every few years the performance/price ratio of hardware doubles. Factors such as increasing processor power, memory capacity, communications speed, and parallel processing will continue to encourage demands for software far beyond our current ability to deliver, particularly if real computing horsepower is to be obtained from this equipment. In comparison, software production has virtually stood still.

Software requirements in the next decade will not simply be more of the same, i.e., larger databases and faster machines. They will include being responsive to the business through the rapid assembly of new and existing software components to meet the specific workplace needs of wherever the business wants to go. They will respond to a movement of power from the developer to the end-user through software component assembly and the new interfaces that give the user more flexible and effective control over system behaviour. All manufacturing disciplines are characterised by the evolution of this movement of power from the developer to the user.

This phenomenal progress in hardware technology coupled with the needs of the business from its IT operation provide the biggest single reason for an effective reuse technology. Object orientation is today the only approach capable of supporting reuse on the scale necessary to enable the major shift in our software production process.

Reuse

We found evidence from practitioners to support the above vision. With regard to reuse metrics, we found high reuse levels (only a small proportion of the component changes each time it is used in different application) in graphics, for example. We did not find sufficient metrics to know what the general reuse levels are for commercial objects, e.g., customer, product,

brand. At the time of the study there were not enough examples.

Failures

Where we saw failures, two factors were involved. First, the analysis had not been done sufficiently well to identify stable domain objects. These emerged iteratively during the "build" process, each iteration sending shock waves through the development team. Continuous prototyping without adequate foundation modelling is not the way. Second, tight deadlines had been agreed to and, apart from the new technology, a whole new management approach had to be invented within those time frames. Careful attention to these factors for beginners is well worthwhile.

REFLECTING THE BUSINESS OF THE ENTERPRISE

Modelling the enterprise

We found sufficient evidence to support the claim that object orientation enables IT to more effectively reflect the business of the enterprise. Object orientation provides a natural model based on classifying the real world, using distinction to partition the world into objects and commonality to group these objects into classes. The ability to directly represent real-world concepts means that a more accurate mapping between the problems and the solution is achieved, which is understandable to both the developer and the user. It differs from other techniques in that a precise definition of business terms arises.

The dynamic nature of the real world is modelled directly and earlier than in other approaches by messages and methods. Up to 80% of important business rules are categorisation and subcategorisation, which encapsulation and inheritance techniques handle well.

Business objectives

Business objectives can be modelled effectively because they must eventually be quantified and translated into a particular state that objects modelling the business must reach.

Business concept interfaces

Object orientation enables the business to be reflected more effectively at the user interface by the use of business object interfaces. Research shows that interface usability and employee effectiveness depends on three factors: appearance (10%), ease of interaction (30%), and that the screen presents concepts that correspond to the users own perception of the business (60%). Users should not be required to think "applications"; they are not a business concept. This is an attractive entry point for the use of object orientation because it allows us to move forward and take new opportunities without major changes to our existing system's legacy. It still requires the user's workplace to be modelled properly and stable objects identified. These objects will only "exist" on front-end intelligent workstations as access to the host will be required to retrieve and store instance data already existing on conventional databases, or to treat whole applications as objects providing services on request. The reality of the business world and its transactions is more accurately reflected by this type of interface because the traditional system "flow of control" has passed from the application to the user. The user now instigates his own interaction with business objects in response to tasks in the order the business determines each day.

Decision Support

Although we did not find supporting evidence, the object-oriented vision should offer an important contribution to decision support. In theory, an object model of an enterprise would respond to events injected by a user for the purpose of "what if" evaluations. Today, decision support and operational needs are met by quite different application types, although they manipulate the same entities.

THE BUSINESS OPPORTUNITY

Competitive edge in the 1990s will depend on the ability of the IT operation to reach out to wherever the business wants to go and to rapidly integrate existing and new functions to meet the local needs of the workplace. There is also a need to make employees more effective by widening the range of activities they can undertake.

This requires a shift to interfaces at which users can easily manipulate familiar business objects that correspond to their own perception of their business world. Current interfaces—despite the sophistication of graphics, multiple displays, etc.—are applications oriented and require the user to know and think "applications," which are not business concepts. Today's concept of a first major delivery followed by years of maintenance must disappear so that all work is seen as part of a continuous, evolving, asset-building production process in which the major part of any new application can be met by the assembly of reusable components. An important independent pressure is coming from the hardware cost revolution (one megabyte of RAM now costs less than two lines of source code). As stated earlier, every few years the performance/price ratio of hardware doubles, encouraging demands for software far beyond our ability to deliver, particularly if real computing horsepower is to be obtained from all this equipment. Object orientation, in our view, is the only technology today that has the potential to address these issues.

continued on page 9...

Globally Optimizing C/C++ Compiler (Version 3.1)

MetaWare, Inc.
2161 Delaware Ave.
Santa Cruz, CA 95060-5706
(408) 429-6382

MetaWare Inc. announced the development of **Globally Optimizing C/C++ Compiler (Version 3.1)**, which will include a full implementation of C++. The package also includes Source Level C/C++ Debugger and MetaWare's Windows 3.0 Applications Development Kit. MetaWare started with its globally optimizing High C technology and has added a full implementation of the C++ language as described by the *C++ Annotated Reference Manual* by Ellis and Stroustrup. "Templates" have been implemented and "Exceptions" are scheduled for inclusion in the next release. For professional developers who are beginning to migrate from C to C++, MetaWare's High C/C++ technology supports incremental "Strengths," a feature that allows the programmer to invoke features of C++ incrementally. You can build upon C and C++ a block at a time by including C and C++ object modules in the final executable.

EZWIN32

MicroWay
P.O. Box 79
Kingston, MA 02364
(508)746-7341

Microway is now shipping **EZWIN32**, their 32-bit Windows Extender. This product provides 32-bit, multilanguage support for the Windows platform, NDP Fortran, NDP C, NDP C++, and NDP Pascal. Microway's protected mode 386 compilers have been extended to take advantage of the Windows environment. As a result, programmers can now simply recompile their 32-bit applications and run them using the Windows 386 enhanced mode. EZWIN32 implements character-based I/O. This makes it possible to create or run a 32-bit character based application in a window. One can develop code under DOS or Windows that runs under Windows without having to become an expert on the 1300-plus Windows API entry points. The NDP compilers for Windows generate programs, procedures, and arrays with no limitation on the size of any segment other than it be less than four gigabytes or the total amount of virtual or real memory in the system. They support the Intel and Cyrix coprocessors and are compatible with other Microway NDP compilers that run under DOS and UNIX on the 386/486/i860.

Green Hills Compiler Family 680X0 Cross Tools

Oasys
One Cranberry Hill
Lexington, MA 02173
(617)862-2002

Oasys will provide the **Green Hills Compiler Family** (C++, C, FORTRAN, and Pascal) and their **680X0 Cross Tools** for embedded development of the new Version 4.0 of Santa Cruz Operation's UNIX operating system. Oasys currently provides the Green Hills Compiler Family and the Oasys Cross Tools for 680X0 on SCO UNIX, Interactive 386 UNIX, DECstations, DEC VAX, IBM RISC System/6000, Sun-4 SPARC, and HP workstations. The Oasys Cross Tools allow users to develop applications on UNIX-based workstations or DEC VAX systems that will execute on 680X0, 88000, or 386 microprocessors.

Oasys' **Green Hills C++** will be integrated with IBM's **AIX Software Development Environment (SDE) WorkBench/6000**, a framework product based on Hewlett-Packard's SoftBench technology. This integration provides C++ compiling and debugging capability for software developers on IBM RISC System/6000 workstations.

Oasys also announced their commitment to support the **DEC OSF/1** operating system. Oasys will provide Green Hills C++ and their 680X0 Cross Tools for embedded development of the DEC OSF/1 operating system. DEC OSF/1 is compliant with the OSF/1 Application Environment Specification, which specifies the interface to support portable applications that will run on a variety of hardware platforms. Oasys will provide their software tools under DEC OSF/1 within six months of the release of DEC OSF/1.

ProtoGen 2.1

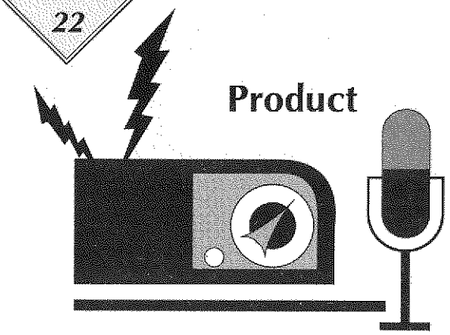
ProtoView Development Co.
353 Georges Rd.
Dayton, NJ 08810
(908)329-8588

ProtoView Development released **ProtoGen 2.1**, a visual program design and code generation tool for Windows application development. Version 2.1 features a completely new menu designer and dialog-linking interface. It also generates code for both ANSI C and Borland C++ ObjectWindows. Additional new features include code generation for Borland's C++ Object Windows library as well as ANSI C, and project file generation for Borland's Turbo C for Windows interactive compiler. ProtoGen 2.1 utilizes Borland's custom control DLL and generates code to access these controls. C++ ProtoView dialog class can be used from within ObjectWindows (OWL) programs to access ProtoView's enhanced dialog subsystem for Windows. A new, powerful menu designer can be used to quickly and easily create menus of any size and complexity. Menu templates can be saved and used again in other applications.

db Vista

Raima Corporation
3245 146th Pl.
SE Bellevue, WA 98007
(206)747-5670
(800)327-2462

Raima Corporation announced that they are changing the nomenclature of **db Vista**, their database management system, to **Raima Data Manager.** This move is in line with the company's new corporate image campaign of relating its product names to their specific technology. The Raima Data Manager DBMS is best known for its high performance in applications development and combined database technology of relational network models. The name change to Raima Data Manager accompanies a new update release, Version 3.21. The update does not present new features, but includes fixes to the current 3.2 version. Raima Data Manager will continue to be marketed as both a stand-alone product and as a full system accompanied by db_Query and db_REVISE. db_Query is an SQL-based query and reporting module; db_REVISE is a sophisticated database restructuring tool.



Product Announcements

Product Announcements is a service to the readers of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY; it is neither a recommendation nor an endorsement of any product discussed.

KBMS Helpdesk Shell

AICorp, Inc.
138 Technology Dr.
Waltham, MA 02254-9748
(617)891-6500

AICorp, Inc. is shipping the early release of the **KBMS Helpdesk Shell**, an integrated problem managment and diagnostic system. The KBMS Helpdesk Shell addresses the needs of a variety of help desk applications including customer support, field service, end-user computing, and vendor technical support. It includes integrated call tracking, problem diagnosis, problem resolution, problem management, natural language query, and reporting functions. The system is designed so that nontechnical help desk staff can build and distribute the knowledge bases used to solve problems. AICorp developed the KBMS Helpdesk Shell in a joint effort with Andersen Consulting, who will provide consulting resources when requested to support help desk customization for AICorp customers. While it can be used without modification, the KBMS Helpdesk Shell is designed to enable users to quickly customize it to meet site-specific requirements. It can also be integrated with existing problem managment systems. In addition, all aspects of its graphical user interface, including menus, controls, and data objects can be modified to meet user needs.

POET

BKS Software
One Kendall Square
Cambridge, MA 02139
(617)621-7047

BKS software announces **POET**, an object oriented database for C++. POET—short for "Persistent Objects and Extended Database Technology"—is an extension to C++. Developers write POET applications using special C++ keywords for class definitions and a runtime system. There is no need to learn a separate database language. Poet is 100% portable across a wide variety of platforms, including DOS, Windows 3.0, Interactive UNIX, HP-UX, and NeXT. Other platforms are either being developed or will be developed based upon demand. Poet has all the qualities of an object-oriented database management system (OODBMS) including persistent storage of objects, searching, selection, sorting, and navigation inside the object space. Poet reads the database structure directly from the C++ class definition. Because the database is expressed in C++ classes, the database can be fully object oriented. All versions of POET use one programming interface and one format for storing data. Because POET is also network compatible, a database server can communicate and exchange objects with clients residing on different operating systems.

Smalltalk/V PM Relational Database Interface

Digitalk, Inc.
9841 Airport Blvd.
Los Angeles, CA 90045
(310)645-1087

Digitalk, Inc. announced availability of its **Smalltalk/V PM Relational Database Interface**. This new extension to Smalltalk/V PM, Digitalk's object-oriented programming system for OS/2 Presentation Manager, provides an easy-to-understand interface to the OS/2 Edition (EE) Database Manager and the Microsoft-Sybase SQL Server. The new database classes implement the application programming interface (API) of EE Database Manager and SQL Server. This allows the programmer access to these database interfaces by creating objects that understand how the underlying database functionality works. The classes include protocols to perform standard SQL operations, send SQL expressions to the database, and create such database objects as `SQLDatabase`, `SQLTable`, and `SQLRow`.

KnowledgePro Windows V2.0 (KPWin)

Knowledge Garden, Inc.
12-8 Technology Dr.
Setuaket, NY 11733
(516)246-5400

Knowledge Garden Inc. announced its **KnowledgePro Windows V2.0 (KPWin)** visual object-oriented programming environment for Windows. The new version adds a comprehensive set of visual design tools, simplified access to dynamic link libraries, and full support for the Windows Multimedia Extension. KPWin offers the unique combination of a full-feature, object-oriented language, hypertext and expert system functions, multimedia support, and ease of use by the novice programmer. More than 80 commands have been added to the new version, for a total of over 300. Users now have the ability to add user-defined tools on a new Customized Tools menu, size and drag Windows objects, and trap mouse-drag events.
Knowledge Garden also announced the **KPWin SVW Video Toolkit**, an add-in toolkit that provides KnowledgePro developers with all the high-level commands necessary to control motion video in real time, using the Super VideoWindows video overlay board from New Media Graphics, Inc.

In a related announcement, Revelation Technologies Inc. of New York, NY, said it has shipped **OpenInsight**, which integrates a custom version of KPWin with Revelation's data-independent application development environment.

Send Product Announcements to Robert Shelton, 1850 Union St., Ste. 1548, San Francisco, CA 94123, Fax (415) 928-3036. Include company name, address, and phone number.

STANDARDS

Report on the Object-Oriented COBOL Task Group

by Megan Adams

This column covers the progress of the group that has the charter of producing a standard specification for Object-Oriented COBOL, the Object-Oriented COBOL Task Group (OOCTG).

In particular, it provides an overview of the functionality included in Version One of the Object-Oriented COBOL feature set. This column reflects the understanding of Hewlett-Packard participants in the OOCTG, and is not intended to be taken as an official Task Group document.

In November of 1989, in response to an HP initiative and growing interest among COBOL vendors and in the COBOL user community, the CODASYL COBOL Committee sponsored a symposium on Object-Oriented COBOL. At this meeting the Object-Oriented COBOL Task Group (OOCTG) was formed, with the charter of drafting a specification for Object-Oriented COBOL. The OOCTG has been meeting quarterly since January 1990 and its participants include both vendors and users (such as IBM, Hewlett-Packard, Hitachi, Realia, Micro Focus, Digital, Sun, Southwestern Bell, and Hewitt Associates.)

At its most recent meeting in December 1991, the OOCTG selected a simple and consistent subset of features from all those that have been investigated for inclusion in Object-Oriented COBOL, and called that subset Version One. This was presented to the OOCTG parent committees (ANSI X3J4 COBOL Committee and the Codasyl COBOL Committee) on January 16, 1992. Version One represents an important milestone in that it includes powerful functionality and provides those interested in Object-Oriented COBOL with an understanding of the OOCTG technical approach. It is also possible that some of the early implementations of Object-Oriented COBOL will use Version One as a guide.

The final technical report submitted by the OOCTG to its parent committees will include some additional features that are not included in Version One; in particular, exception handling.

GOALS OF OBJECT-ORIENTED COBOL

Before discussing specific features in Version One, the goals that have guided the work of the OOCTG will be outlined. These goals are consistent with goals of other object-oriented languages, and are driven by the overriding goal of assisting programmers in handling the problem of software complexity.

The first goal for Object-Oriented COBOL was to improve the modeling of real world problems in software. Normally, in COBOL, a bank account problem is translated into a set of logical operations distributed among programs and files. Object-Oriented COBOL supports the approach of modeling the bank account objects and their associated operations directly in the Object-Oriented COBOL application.

A second goal for Object-Oriented COBOL was to provide support for modularity. Objects should be accessed through interfaces, so that changes behind the interface need not affect clients of the objects. This approach is supported in Version One. For example, clients of bank account objects need not be affected by changes in bank account data layout or bank account service implementation, as long as these changes do not affect the interface to bank account objects. This approach eases maintenance and improves reliability.

The third goal for Object-Oriented COBOL was to support software reuse. Reusability of software components is potentially the most important single factor in increasing software productivity. Version One of Ob-

Megan Adams has been a software engineer on the COBOL compiler project in the California Language Lab for seven years. She has been HP's primary member of the CODASYL COBOL Committee, the standards body that designs new features for COBOL and an alternate on the ANSI X3J4 Committee, which adopts and publishes ANSI Standard COBOL. Adams also serves as Vice-chair of the Object-Oriented COBOL Task Group.

ject-Oriented COBOL supports this capability. For example, the definition of bank account objects can be reused when constructing definitions of other classes of objects, such as checking account objects.

Another goal for Object-Oriented COBOL was to support incremental development. Existing definitions of classes of objects must be easily specialized and extended as new requirements arise. Version One supports this capability. The Object-Oriented COBOL class-defining bank account objects is easily extensible to provide a definition of credit card objects.

Other goals for Object-Oriented COBOL relate to the specific characteristics and priorities of the COBOL user community. These include compatibility with existing COBOL, ease of learning, and ease of use.

FEATURES OF VERSION ONE OF OBJECT-ORIENTED COBOL

Version One defines a set of basic features, which are listed below and then discussed in turn:

- Object
- Class
- Inheritance
- Interface
- Object Reference
- Conformance
- Generic Code
- Factory
- Object Lifecycle

Object

The basic construct of Object-Oriented COBOL is the *object*, which groups together object data and object services, called methods. CUSTOMER-ACCOUNT, for example, is an Object-Oriented COBOL object containing BALANCE and ACCOUNT-NUMBER object data, and the object services DEPOSIT and WITHDRAW, which operate on the object data. Objects allow applications to directly model the problem domain. Objects in Object-Oriented COBOL can be both dynamically allocated by users and statically pre-allocated by the compiler.

Class

The *class* in Object-Oriented COBOL defines the implementation of a class of objects. This includes defining the object data formats and the object service implementations, or method bodies. For example, the implementation of CUSTOMER-ACCOUNT object is defined by the class BANK-ACCOUNT. All BANK-ACCOUNT objects share the same services (DEPOSIT and WITHDRAW), and share the same object data formats for BALANCE and ACCOUNT-NUMBER.

Inheritance

Inheritance in Object-Oriented COBOL is a mechanism for code reuse. It is especially useful for extending and specializing existing functionality. A class CHECKING-ACCOUNT can inherit the services DEPOSIT and WITHDRAW from BANK-ACCOUNT, reusing this functionality, and add the service PROCESS-CHECK, extending functionality.

Interface

All objects have *interfaces*. The interface to an object consists of a description of the services that the object supports. The object interface CUSTOMER-ACCOUNT describes the services DEPOSIT and WITHDRAW. Object interfaces are separate constructs from class definitions, because the interface describes the services available for an object, while the class describes the implementation of an object. The object interface for CUSTOMER-ACCOUNT describes the parameters and return values for DEPOSIT and WITHDRAW, but does not include the code for DEPOSIT and WITHDRAW.

Object Reference

In Object-Oriented COBOL *object references* are used to access the services of dynamically allocated objects. Object references generally specify the interface supported by the object they are referencing. For example, the reference CUST-ACCT-REF can be declared to reference an object of the BANK-ACCOUNT interface. A client can then access the services DEPOSIT and WITHDRAW through this reference, by using a new COBOL verb, INVOKE.

Conformance

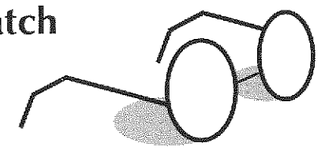
Conformance in Object-Oriented COBOL is based on interfaces. If one interface, CHECKING-ACCOUNT, contains descriptions of all the services in another interface, BANK-ACCOUNT, then CHECKING-ACCOUNT conforms to BANK-ACCOUNT. (The full conformance rules are not presented here.)

The important point regarding conformance is that if CHECKING-ACCOUNT conforms to BANK-ACCOUNT, a CHECKING-ACCOUNT object can be used wherever a BANK-ACCOUNT is expected. This is because CHECKING-ACCOUNT supports all the services that BANK-ACCOUNT supports. This has the important consequence that if a new class such as CHECKING-ACCOUNT is added to a system, old code that expects BANK-ACCOUNT objects will continue to work without change. This makes it easier to incrementally modify systems to meet changing requirements. Conformance relationships are fully checked by the compiler.

Generic Code

Conformance relationships support *generic code*. Generic

Book Watch



The "Book Watch" column does not contain book reviews. These listings are abstacted from press releases provided by the publishers, and no endorsement is implied. Please send announcements to the Editor: Robert Shelton, 1850 Union Street, Suite 1584, San Fransisco, CA 94123; Fax: (415) 928-3036.

Borland C++ Programming for Windows

Bantam Books 666 Fifth Ave.
New York, NY 10103
(212)492-9545

Bantam Computer Books has added *Borland C++ Programming for Windows* by Peter Norton and Paul Yao to the Borland Bantam Series of official books. Written for the intermediate -to-advanced C programmer, *Borland C++ Programming for Windows* covers programming techniques, the ObjectWindows library, requisite information on Windows architecture, memory management, Graphics Device Interface (GDI), compatibility issues, and Windows programming tools. Peter Norton is the author of the popular Norton Utilities software package for IBM and compatible PCs, as well as the Norton Commander program and Norton Utilities for the Macintosh. Paul Yao is president of International Systems Design and has authored several books, including coauthoring Bantam's *Peter Norton's Windows 3.0 Power Programming Techniques.*

...continued from page 20

One might expect it to become unmanageable with large scale problems. We have found this to be true in part.".

It is unclear from the authors' presentation how this approach qualifies as object-oriented. The Information Structure Diagram does include super-type/sub-type, but I don't believe this is sufficient to handle inheritance issues that are otherwise uncovered. Left as an exercise for the reader is figuring out how to capture business rules, preconditions, and postconditions that are critical to preserving the semantics of the object, not to mention the business or subject being modeled. Granted, these are process questions, but the thing that differentiates object-oriented from structured analysis is the de-emphasis of implementation issues (stored vs. computed data) until much later in the cycle.

Before deciding to purchase this book, ask yourself three questions. First, would you describe your projects as small? Second, can you purchase a Computer Aided Software/Systems Engineering (CASE) tool that supports this methodology? Last, until what point in the development lifecycle will the diagrams and documentation be useful? To get maximum benefit from any technique and supporting methodology, such as Shlaer-Mellor's, the answer to either of the first two questions must be "yes," and the answer to the last must be "throughout."

In my opinion the book falls far short of this standard because it does not provide sufficient information to live up to the claims made on the back cover: "This is the only book that treats the subject in enough depth to tell an individual practitioner how to:

- collect information and expert knowledge about the problem at hand
- transform that knowledge into well-defined objects, attributes and relationships
- fully incorporate the resulting information model into the software development process." ≡≡

BOOK REVIEW ≡≡

OBJECT ORIENTED SYSTEMS ANALYSIS, MODELING THE WORLD IN DATA

reviewed by Michael Fuller

by Sally Shlaer and Stephen J. Mellor

OBJECT ORIENTED SYSTEMS ANALYSIS by Sally Shlaer and Stephen Mellor is another in the recent spate of books updating the Yourdon structured analysis/structured design approach.

The book adopts a popular press writing style and picture book format to introduce newcomers to the field of designing software.

It is a remarkably short book, totaling just 149 pages. The main text of 108 pages presents relational theory, semantic constructs, a new diagramming style, interview techniques, and a format for supporting documents. As promised by the back cover, there are lots of cartoon drawings to clarify the text. The remaining 34 pages consist of the table of contents, an introduction, two fully worked examples, bibliography, and index.

I regularly use several of the techniques presented in this book, and agree with the authors that they are effective at capturing and managing the complexity inherent in developing business systems. In sum, Shlaer and Mellor prescribe seven documents or techniques to be developed during object-oriented systems analysis:

1. *The Information Structure Diagram* is derived from the Entity Relationship models proposed by Chen (1977), Tsiritzis and Lochovsky (1982), and Martin (1985). This version lists the name of the object followed by the attributes that describe it. Attributes are marked with an asterisk to indicate participation in the primary key or a "dot" to indicate participation in an alternate key. Relationships are drawn using the "arrowhead" notion to indicate multiplicity, are named in both directions, and are marked with a "C" to indicate conditionality.
2. *The Overview Information* is a simplified version of the Information Structure Diagram with the attribute names and associated key indicators removed.
3. *The Object Specification* Document is a textual description of the entire model. Included is a definition of the object and each of its attributes, declaration of primary and alternate keys, and the domain and business function of each attribute.
4. *The Relationship Specification* Document is a textual description of each relationship pair (from/to each object). Included in the definition is the multiplicity, conditionality, what business policy the relationship is enforcing and why, and what attributes are implementing the relationship.
5. *The Short Form Document* is a data dictionary type listing of the objects with attributes and relationship phrases (Region CONTAINS Common Block (1:Mc)).
6. *The State Transition Diagram* is used to document the lifecycle of the information model objects. This is intended to integrate the data aspects with the event/action pairs that actually cause the data to be stored or changed.
7. *The Data Flow Diagram* has not been modified from the technique proposed by Tom deMarco (1977) with the exception that data stores are replaced by objects.

My concern is that the authors imply that these techniques are so powerful that managing the complexity of analysis is reduced to a simple, though tedious, documentation exercise. My experience is that these techniques encourage you to engage in a comprehensive analysis and the amount of information collected can be overwhelming for a project of meaningful size. Maintaining the drawings and supporting documentation by hand is impractical. The authors seem to share this concern, as they state "The Information Structure Diagram (ERD) contains a great deal of information:

continued on page 21...

Michael Fuller is an information engineering consultant, and has developed large scale, distributed applications with a variety of technologies, including Eiffel. He can be reached at (415) 928-7067.

code stresses the logical functionality over the service implementation, because the same logical functionality can be bound to a variety of service implementations.

For example, suppose CHECKING-ACCOUNT reimplements the WITHDRAW method it inherited from BANK-ACCOUNT, to impose a transaction fee on each withdrawal. The service WITHDRAW has the same interface in CHECKING-ACCOUNT and BANK-ACCOUNT classes, but different implementations. Because CHECKING-ACCOUNT conforms to BANK-ACCOUNT, the service WITHDRAW can be requested of both a CHECKING-ACCOUNT and BANK-ACCOUNT object, and in fact the code invoking this service does not need to know which kind of object it is dealing with. In this case, the WITHDRAW request is generic code, and the appropriate WITHDRAW implementation will be automatically dynamically bound to the request at runtime.

Factory

The class definition contains an object portion, described above, which defines object code and data, and a *factory* portion, which contains factory code and data.

The factory portion of the class definition defines the services and data of a factory object, associated with that class. The factory object always exists for the duration of the run-unit (unlike objects, which can be dynamically allocated and destroyed), and there is always one per class. This allows the factory object to support two important functionalities. First, factory data are shared by all objects of a class. (Object data are unique to each object.) In the BANK-ACCOUNT system, such shared data might be useful for holding interest rates associated with classes of accounts, such as interest on a savings account. All objects of the savings account class would share the interest rate data defined for this kind of account.

Factories also all support the service of new object creation. Requests for new objects go to the factory, which actually allocates the memory for the new object.

Object Lifecycle

The *object lifecycle* consists of object creation, initialization, finalization, and destruction. The user interface to the lifecycle consists of one verb, INITIALIZE, which triggers both creation and initialization of an object. Users do not need to worry about finalization or destruction; these operations are carried out after automatic memory management determines that objects are no longer referenced.

While the user interface to the object lifecycle is simple, class designers have considerable power available in defining the implementation of the various phases of the lifecycle. The users see only INITIALIZE, but the class designers can define methods that implement creation, initialization, finalization, and destruction. If these are not defined in the class, defaults are provided by the system.

SUMMARY

Version One of Object-Oriented COBOL contains some features that have not been touched on here, due to the brevity of this column. The intent of this summary is to provide a simple overview, to impart an idea of the power of the feature set to those unfamiliar with the work of the OOCTG, and to communicate a general sense of the technical direction of the OOCTG. ≡≡

USERS' ASSESMENT...continued from page 6

NO PANACEA

It is clear that these benefits are unlikely to be achieved without experienced assistance and investment in training. New management problems appear. Changes in culture and the need to reorganise are common in most IT organisation. Management of software as an asset (especially at the analysis and design level) will be a novel experience for most enterprises. We concluded there are no methodology, CASE, or database products sufficiently mature to justify immediate migration of any large-scale IT function. From a commercial systems viewpoint the available products and techniques can currently be used for small to medium applications, for front-end work (new user interfaces and integration of large existing systems to meet the needs of specific workplaces), and for enterprise modelling. Nevertheless, there have been several recent examples (e.g., the automobile industry) of its use for large-scale enterprise-critical applications either to gain a competitive edge or because a competitor has adopted the technology.

The way forward is to explore the technology and its management on pilot applications, front-end work, and enterprise modelling. On a broader front, a proposal for a further year's work by the OIG is circulating for approval. The work is focussed on helping members to get started, developing method, technique, and organisation based on sharing real project experience, and influencing suppliers to respond to our needs and priorities. ≡≡

Norman Plant is Chairman and a founder of the Object Interest Group, and a consultant specializing in the introduction of object technology into large IT organizations. He recently retired as a senior manager in the IT division of British Airways, after 25 years service to the company. He can be reached at +44-252-836315, in the United Kingdom.

APPLICATION STRATEGIES

Discardable software systems in the object age

by Frank Douglas

The pyramids were built to perform a single function. They were used by the Pharaohs once. Cortez burned ships built to be reused after a single trip.

In the 1940s, four years of the most intensive research in history resulted in the most expensive single-use product imaginable: the atomic bomb.

Discardable things are not new in science, technology, or history. Our beaches and highways are littered with things like razor blades and tin cans: all built as if to endure, but all destined to be thrown away. Software systems are built from electrons; they vanish at the snap of a switch. But, methods for design and construction of software resemble those used to create Styrofoam cups. The software systems they evolve may be destined to become electronic litter.

Object-oriented technology is one of a host of forces changing the field of commercial application development. For example, "business re-engineering," redefining the basic business process and goals for a company, is an influence that makes the ability to change how a company does business an important competitive requirement. Similarly, application technology is being made obsolete nearly every quarter. As business processes are changed, application software must change to provide tools and mechanisms for new opportunities. As mainframe applications are supplanted by standalone, character-based mini-applications, and as these, in turn, are supplanted by graphic-based processes, application software must be modified, discarded, or migrated. There is no stability today; there is likely to be less stability in the future.

Economic trends are increasing the competitive pressures in the marketplace as large, once invincible companies experience reversals of fortune. All of these forces mean just one thing: The rules for doing business are changing radically. The rules for building applications will also change.

Businesses that cannot change will fail. If developers of business applications cannot adapt to the new, more volatile business environment, they themselves will be discarded.

One of the first aspects of application development to be eliminated is the standard 2 x 2 system development cycle ("two years, two million dollars"). It is unsuited to current business pressures. In fact, methods of Rapid Systems Development are proliferating in an attempt to come to grips with the problems created by continual change in business and technology. These methodologies are only a stopgap.

The problem is more fundamental. The concept of fixed-stage, invariant-sequence development patterns is inadequate; it may even be true that the entire concept of the application lifecycle is maladaptive. Given constant changes in the technological base for applications, the stimulated changes induced by a "business re-engineering" industry, and the pressures generated by economic decline, major fixed-stage, invariant-sequence methods will produce systems likely to be obsolete well before their requirements are fixed and documented. The lifecycle concept, employing an assumption in the value of software as an "asset," will compel extraordinary efforts to "salvage" and "refit" these assets: an impossible and self-defeating task. The lifecycle concept may be better suited to a stagnant or declining technology than to one that is changing its foundation assumptions many times a year.

One way to cope with these changes is to build software so cheaply and so fast that it can be thrown away when its use has ended. It has no value as an asset, any more than a paper towel does.

Discardable or "single shot" applications are not new. It is always a treat to watch "old-time" programmers react to the phrase, "discardable." They respond that that's how they built their current A/P system 20 years ago. Still others comment, "Sounds like APL to me." Nearly

Frank Douglas is a consultant in the New York area.

DOWN UNDER

The Australian object-oriented scene

by Brian Henderson-Sellers

I would like to reflect on some of the more exciting object-oriented events around the country during the previous 12 months. There are perhaps two perspectives to report upon: national and local activities.

Our most recent national activity, TOOLS 6, was held in Sydney in December and attracted a substantial audience from across the country as well as many international visitors. As with all TOOLS conferences, the format is two days of tutorials followed by the technical program, including keynote speakers. Although it is possible to attend either portion, it was encouraging to see that most delegates attended the full conference. Although, as the Tutorials Chairman for this conference, I should admit to bias, my impression from talking widely to delegates was that all the tutorials and technical paper sessions were very well appreciated. Probably the biggest attendances were for the tutorials by Adele Goldberg and Kenny Rubin (project management), Roger Duke (formal methods), and my own (analysis and design). That the enthusiasm in Australia is biased towards management and analysis and design methodologies reflects the high degree of industrial/commercial interest in getting down to real implementations. I find that extremely encouraging. In a similar vein, the conference closed with a panel of international and national representatives responding to questions from the audience on the real issues of commercial adoption of O-O. It is also intriguing that the use of formal methods within the O-O development paradigm is so well supported. Again, I take that as an indication of the seriousness and enthusiasm for O-O within Australian business.

The other national meeting of interest was a much smaller workshop to which all Australian academics currently researching into O-O were invited. This was highly successful in bringing together people from across the continent who had previously been only names or voices on the telephone to each other. The meeting has certainly fostered good relationships across a country that is as huge as North America.

For the majority of practitioners, national meetings serve a limited purpose, as they are held infrequently. Local, state-based interest groups provide a more ongoing service. As I write, there are now not only Special Interest Groups (SIGs) of the Australian Society in Sydney (New South Wales) and Melbourne (Victoria), about which you heard in the last Down Under column, but also a new group has recently started in Brisbane (Queensland), and a decision was taken in September to look into the possibility of organizing an ACS O-O SIG in Perth (Western Australia). In addition, many commercial organizations have held one- to five-day seminars or courses for detailed consideration of selected aspects of object technology. Speakers at these have been both national and international protagonists for object technology.

All in all, despite some O-O projects being canceled purely from business reasons (Australia also has a fairly deep recession), the general outlook for O-O in Australian business is optimistic. ≡≡

B. Henderson-Sellers is with the School of Information Systems, University of New South Wales. He can be reached at brianhs@usage.csd.oz.au.

doesn't mean that the dream was wrong or that reality can't get there; object modeling is an important technical breakthrough in completing the support for a model of the business that is active, flexible, and complete. At Intellicorp, we have the knowledge and experience to build very powerful modeling and design tools. We need to partner with the companies that have been in the traditional CASE arena so we can share our representation technology with them. They, in turn, can share with us their knowledge of large team programming projects, project management, documentation, repository interfaces, and model management. A lot of value has already been delivered in these areas in the existing CASE market, so we will work with partners to obtain the pieces of the puzzle that we ourselves don't complete. Working with James Martin is the first and a very critical step in building this strategy. Realistically, one company of our size cannot do it all. Partnering with two or three experts in complementing areas can bring a better solution to market.

HOTLINE: This must have been the basis for your working with KnowledgeWare?

BRANSCOMB: Well, that wasn't exactly "working with"—that was getting married! What we're doing now is more like living with our partners. Successful partnering is critical in this market. The winners in today's tools market are those who don't just offer a good line of tools, but also deliver the partnerships and services so the customer gets the complete package.

HOTLINE: Given James Martin's direction with object-oriented planning, how will Intellicorp provide tooling or support for business re-engineering?

BRANSCOMB: There's a link between this strategic level and the IS organization, but we're only beginning to scratch the surface. Today we're focused on helping people build better computer applications, leaving strategy up to the executives. If you look a couple of years ahead, the bridge between these will be very solid and successful. My rule of thumb is "only one new thing at a time." It's new enough to try to put object technology to work on traditional IS problems. If we can make the technology accessible, there are other people who can help embrace the strategic uses of these tools. Martin is clearly leading the way. Our biggest challenge is boiling object technology down to its essence and packaging it in as approachable a way as possible.

HOTLINE: One way that object technology is expected to contribute to business re-engineering is through providing tools for business visualization. How do you expect this to develop?

BRANSCOMB: Business visualization is about capturing business requirements more directly in software. I think it will evolve to simulation, but today being able to allow the user or systems analyst to actually see a graphic metaphor for the business requirements during the construction process is critical—to extend the vision of the business and its goals as far down as possible into the software development lifecycle. Traditional procedural programming languages often represent more of a barrier than a bond between the end user's business model and what the programmer is trying to produce. We think that visualization is important, particularly in the later stages of the software lifecycle—especially in the construction phase. You'll need that information when you go to maintain or change your application. We have three techniques for keeping as much of the business vision as possible alive in the later stages of construction:

1. maintain the structure of the business in objects and relationships
2. maintain the process and policies of the business in methods and procedures
3. maintain the decisions and strategies of the business in rules and goals

By supporting at the language level these types of semantics, we not only enable higher-level graphic metaphors for modeling and simulating the business for managers, but we give the implementors the language-level tools that have within them the essence of that higher-level view. So, in some sense, with objects the specifications are directly executable because that's the level at which they are captured.

HOTLINE: What's the bottom line on object-oriented CASE products?

BRANSCOMB: If it can't contribute to the strategy of the company, it's not worth buying. IS wants to know how to get better value out of what they've already bought. The IS customer is not looking to replace what they have with something entirely new. The best way to introduce something new is to give them a bridge from what they've already bought. ≡≡

every mainframe programmer or administrator has a stockpile of discardable applications. These are small processes written in an easily manipulated format such as SQL or REXX. They perform a single function, then are discarded. They are not discarded in the sense of being thrown away, but they are usually consigned to a software attic awaiting recall to service when the identical need arises.

In the microcomputer realm, single-use applications are extremely common. Nearly every microcomputer environment possesses a "language" intended to automate some part of the process. Word processors and spreadsheets have macro languages. Other environments such as Microsoft Windows provide a keystroke recorder. A user can develop specialized functions which will never need alteration, but much of the time the function is highly specific, used for a limited period of time, then retired. If it is used again, it will have to be modified.

Even if ample precedents exist, there is virtually no knowledge concerning how, why, or when limited-use applications can be fabricated. The very idea of building applications to be used and discarded is disreputable today. Even though discardable applications lack recognition as a legitimate endeavor, they may well be the "silver bullet" for the application programmer to use to quell the turbulence being produced by changes in the 1990s. The emergence of specialized tools particularly well suited for creating discardable applications is an interesting development in this regard. For example, Borland's ObjectVision allows either a programmer or an end user to develop a sophisticated data entry system within a few hours.

Nor is it surprising that object-oriented development environments such as Smalltalk or Actor seem to be particularly well designed for developing and managing discardable, limited-use application systems. Smalltalk/V was used effectively to create such an application in a recent "Development Shoot-out" conducted by PC WEEK. The assignment was to develop a "single use" application quickly. Of course, the goal of the competition was to determine which platform was the most effective in developing new applications. But the stated purpose of the competition may turn out to be secondary to the circumstances and requirements presented to the participants in the "shoot-out."

A "throw-away" application developed within the span of a few hours or a few days is well suited to the kind of volatile business environment that is starting to emerge. The application can be composed quickly in response to sudden changes in business requirements. Because the system must only accommodate a limited number of business cases, the code does not have to incorporate tests for totally unexpected variations, nor must it manage complex error conditions.

One fact must be understood from the outset: An application developed within a period of a few days, or a few hours, will not have the generality or the robustness of one that has been carefully designed, developed, and implemented. Programs on the perishable end of the continuum will accomplish very specific purposes. There will be no opportunity to design to a more general case. The usefulness of such a system expires when that specific purpose changes.

The system will be implemented by the developer interactively and adjusted until it produces the correct effect. Whenever it fails to function correctly, it can be changed in a highly specific way—the kind of change we call a "patch" in a more enduring type of system. The high degree of specificity means that design, testing, and implementation are simplified. Much of the system's cost-producing components are simply removed: They are not needed. When the application is no longer appropriate, it is discarded.

> ***"The very idea of building applications to be used and discarded is disreputable today."***

What a "single-shot" application sacrifices in robustness it more than reclaims in other areas. One very important benefit is that totally new classes of business functions can be given computer support. Typically, a business process must be repetitive to qualify for computerization. The process must recur in an almost identical form to allow a software designer to identify the requirement with any certainty. It must repeat itself sufficiently frequently to justify the expense of designing, developing, and implementing a system. These processes are associated with fairly well-established business systems. They are also associated with larger businesses.

The requirements for "critical mass" of repetition are almost certainly absent in two important cases, both of which will become progressively more important in the next decade. In the first case, startup companies, having limited assets, attempt to exploit poorly understood opportunities that yield much smaller profits at higher risks. The second case is similar: Existing companies undertake a one-time initiative to meet com-

petitive demand. For example, a company might develop a one-time sales campaign. In fact, the PC Week shoot-out represented a variant on such a speculative one-time effort.

Neither of these situations can justify a major systems development effort. In fact, it is likely that both would forego computerization during the most critical phases of their life. These two scenarios are becoming increasingly common. The need for computer support without a concomitant need for commitment to major assets is what will motivate limited-use application development.



Opposing the belief in the ultimate law and order of completely generalized systems may seem to be heresy, but these beliefs are pure habit.



But if there is a need for single-shot applications, there is a significant force working against it: the force of habit. It is astonishing that in an area as young as system development, there are so many binding habits and customs. The standards of evaluation are based upon an almost clinical level of repetition compulsion. Most professionals working in this field are infected with this need for perfection.

The force of the opposing habits can be seen even in situations meant for disposability, such as in the case of simple edit macros used to correct a long listing. Most programmers will try to obtain maximum generality and reusability. Rather than trying to develop a simple macro to add a single line to a file after every fifth line, a typical programmer will try to develop a macro to add *n* lines after every *m*th line. Nearly every programmer has filled libraries with such macros.

There is a belief that systematic conception and careful design is synonymous with the fixed-stage, invariant sequence methods. Designers generally believe that any explicit event is an example of a more general "law." It is that law that must be codified and automated. The event is trivial, even misleading. This belief has been encouraged by those who are attempting to rid commercial systems of carelessness and even incompetence. Opposing the belief in the ultimate law and order of completely generalized systems may seem to be heresy, but these beliefs are pure habit.

It is a habit that is difficult to replace, particularly in the case of discardable applications development. The designer of such systems will require a mastery of higher forms of reasoning, including the ability to distinguish patterns indicating the operation of a more general case from those in which the event manifests nothing more than itself. Learning to "cut the Gordian Knot" will be a difficult task for a generation of application designers taught to reason in a single way.

Discriminating between situations that encourage disposability and those that demand reusability is the first step in breaking the habit and introducing such applications into general use. Some situations are quite simple: end user applications involving queries of a database are a simple example of candidates for discardable systems. It is usually better to develop parameter-driven queries of a database that can be replaced as needed than to develop a vast set of programmatic static queries. On the other hand, the database management system had better be developed as an enduring system with extremely high levels of reusability.

Software manufacturers seem to have recognized that a need for single-use or limited-use applications exists. End users have generally presented their business requirements in this specific manner. Application developers need to develop applications tailored to specific circumstances. It may be that very low-cost, highly specific application systems offer the best application development methodology for the coming decade.

We are in an era in which the computer seems indispensable to business function. This may only be true because large businesses have found it expedient to replace highly repetitive functions with a few lines of instruction to a computer. It will not be true when an explosion of small businesses may find the computer-based application an obstacle to their success. It is quite likely in this economic and technological climate that, if no alternative to the fixed, invariant stage type of development pattern is found, application software development may share the fate of other endeavors. Commercial application development may constitute the foundation of economic resurgence, or it may become a quaint archaeological remnant of an earlier age.

In retrospect, "The different accidents of life are not so changeable as the feelings of human nature. I had worked hard for nearly two years, for the sole purpose of infusing life into an inanimate body. For this I had deprived myself of rest and health. I had desired it with an ardor that far exceeded moderation; but now that I had finished, the beauty of the dream vanished, and breathless horror and disgust filled my heart."

—Mary Shelley, *Frankenstein* ≡≡

BRANSCOMB: The customer for object technology is not just IS, because there are industries that do not support the traditional IS structure. Our biggest penetration is in transportation, telecommunications, and integrated manufacturing. In the manufacturing market, the engineering group often has its own information support group—it may not be the same IS department that handles the general ledger. In telecommunications, such as the European PTTs and the U.S. Bell Operating Companies, the IS organization is considered a traditional IS organization in structure, but they are much more heavily involved in operations. They deal with backbone networks and switching equipment. Intellicorp's customer would not be the advanced development or research department; not the information center or decision-support groups that have VM systems installed and large numbers of PCs running Lotus; and not end users except to the extent that the end users are also programmers or engineers.

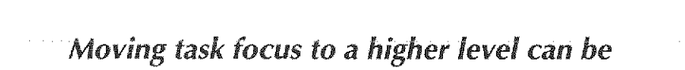
HOTLINE: Much of your emphasis is on evolution from and integration with today's technology, but we still have to achieve a shift of perspective to get from procedural technologies to object-oriented. How do you attack this?

BRANSCOMB: We're learning as we go along, but what we've seen in the accounts that have embraced CASE is that it's often difficult to take a 20-year C or COBOL veteran and ask them to turn their head inside out and think about encapsulated objects, dynamic typing, and message passing. Moving task focus to a higher level can be tough, but usually when these folk have a chance to work with the tools they find themselves liberated from the more grungy aspects of being a programmer. In any organization, there are people who live and die for doing very low-level memory optimizations on their programs. Those people might be customers, but they'll like PROKAPPA because they can get down and dirty in a traditional way. They'll value the hybrid aspects. On the other end of the spectrum are the CASE users, systems analysts, and 4GL users who are focused first on the application requirements and business strategy. For them programming is only a secondary concern, and PROKAPPA represents a much more direct link to the business problem at hand. If these analysts have had experience with ER diagramming or information modeling, they'll find the transition to objects and relations a very natural one. Our goal with PROKAPPA is to provide one tool that can embrace both audiences; therefore we provide both levels of interface: the down and dirty C interpreter and tools, and the high-level graphical modeling interface.

HOTLINE: If I understand correctly, you're saying there is still a need for close-to-the-machine programming on some applications, and that object-oriented doesn't necessarily mean abstraction without the ability to touch the bits and bytes?



Moving task focus to a higher level can be tough, but usually when these folks have a chance to work with the tools they find themselves liberated from the more grungy aspects of being a programmer.



BRANSCOMB: Yes, it's important once your application is functionally complete to be able to bring in the programming wizard to optimize for the particular memory management, operating system, hardware, etc. The trick is not to require the primary applications developer to deal with the lower-level aspects. Once the business application has been fully developed by the business application developers, it can be optimized where and when required by someone who specialized in the machine environment.

HOTLINE: How will the object technology industry be changed by the introduction of information engineering?

BRANSCOMB: The introduction of information engineering is the healthiest thing that could have happened to our industry, because it will put our best minds to work on realistic, tough problems that represent the greatest potential value of the technology. It's a powerful and very welcome dose of reality. There will be a lot of learning on both sides. The winning companies will be those that learn along with their customers how to put this technology to work.

HOTLINE: From our discussion, I gather that you see object technology for the IS user moving toward model-driven development. This raises the issues of model management and repositories that the data administration and IE folks have been struggling with for several years. How will you handle these issues?

BRANSCOMB: Like so many other revolutions in the software industry, sometimes the dream is greater than the reality. This has been true with CASE, but it

James Martin on CASE tool development, perhaps you could elaborate on your vision for CASE. Where is the technology going, and what do objects bring to the party?

BRANSCOMB: If we look back over the last 15 years of evolution in software development techniques, we've done a good job of capturing the information model—the structure of the business. We can access all of the important measures of business productivity. We can locate all the boxes in the warehouse; locate and name all the customers. We can store all the data and capture the structure of the company. What we have not done a good job of—where software technology is still in the stone age—is capturing the behavior and process of the internal workings of the company. Historically, software has not easily represented the goals, assertions, business policy, and procedures of our organizations. We haven't been able to captures the behaviors in a way that easily marries with the structure captured in the information model. What you'll see in the next generation of CASE is much more precise attention to ways of describing and representing in software the strategy and policy of the company's managers. Object CASE will provide for capturing business behavior that is itself a very fluid thing. The structure is fairly stable and repetitive over time. Management, policies, and goals tend to be fluid over time. Modeling events, diagramming business policies and rules, and incorporating that into computer models of the business will probably be areas where you'll see the greatest innovation.

HOTLINE: How does this work for a company that has delivered construction tools for the last ten years?

BRANSCOMB: Intellicorp is primarily a design- and construction-focused company but, applying the theory of garbage-in, garbage-out, it's very important that there be good graphic metaphors and modeling tools for capturing the essence of a business strategy and it's relationship to information in the company. The upstream models must be able to drive the downstream construction of the computer application.

HOTLINE: I've noticed a tendency in the O-O community to reject everything that smells of relational or information modeling. There's a pronounced tendency to say "that's not really object oriented," which sounds to me like "that's not what I invented." Is the O-O community throwing out the baby with the bath water?

BRANSCOMB: Yes, and there's no need to throw the baby out! We've seen very powerful value accrued with entity-relationship diagramming and relational database technology. The combination of ER diagramming and tools for translating that into relational database implementations is valuable. It would be a mistake to throw out what's been good about the tools people have installed for data management and modeling. Information modeling techniques are generally viewed as adequate. The problems with that technology result from how that information model resolves itself with or is bridged to the business logic and interface parts of the business applications. That's where we've fallen down on the job. That's where object methods and programming hold the promise of extending what we've already invested in information models out to the interface and process logic. I don't think we have to move very far afield from ER and RDBMS to incorporate the benefits of OOPS. The real impact is seen is on the process logic side.

HOTLINE: You're advocating an extension to technology that IS already has and knows how to use?

BRANSCOMB: Yes. You get the structure with ER modeling. To get the behavior and the strategy, you need something new. The data is what the behavior and the strategy operate against. It's on that side that OOPS brings the most to the party. The pieces that have not served us well are the data flow diagrams, pseudocode, structure charts, etc. Where you want to store policy is in conjunction with those pieces of the business to which it relates. It seems natural to want a close relationship between the what and the how. Today we bifurcate those two things, and ask IS to go down two different paths to be able to express the goals of the business. Many times the goals and strategy get lost or stripped out, and you end up with only the data. Lost are the reasons why the data's important. If anything, we have too much data. Most companies are facing the problem of a surplus of data, and a deficit of tools for extracting meaning from that data and putting it to work.

HOTLINE: From the way some of these issues get pushed aside by many of our colleagues in the object technology community, I wonder if they have never faced these problems of business information management?

BRANSCOMB: I think that's the case. Some of my colleagues take kind of a purist view. They see a software metaphor that might be valuable, but minimize what is underneath. The software metaphor is just the tip of the iceberg. Project management, data integrity, resource utilization, team programming, and data management are all part of the ancillary infrastructure that's required to make a large-scale IS project successful.

HOTLINE: You've spoken of IS as your target customer, but what about non-IS organizations? Will they benefit from object CASE?

INTERVIEW ≡≡

Bringing object technology to MIS

K.C. Branscomb interviewed by Robert Shelton

Intellicorp purveys object-oriented visual development tools that build client-server applications—in standard ANSI C program code—that can be integrated with existing applications and are connectable with standard relational database management systems.

After the recent nonmerger with KnowledgeWare, Intellicorp's new CEO K.C. Branscomb is focusing the firm on integration of objects with the standards, technology, and data-centered engineering methods of today's corporate information systems (IS) organizations. Ms. Branscomb's crisp understanding of cooperation, compatibility, standards, and building on existing information-sharing mechanisms represents a fundamentally different approach than has been taken so far by other vendors of CASE and methods for object technology (OT). As Intellicorp breaks away from its past as a vendor of AI tools, Ms. Branscomb is betting the company on melding objects and information engineering as the delivery vehicle for driving objects into mainstream IS culture. As Branscomb observes, this is no mean trick!

–Editor's note

HOTLINE: What is the focus of Intellicorp's strategy, and what brings you to the object-oriented technology marketplace?

BRANSCOMB: Intellicorp is putting object technology to work for corporate information systems organizations in a meaningful and pragmatic way by supporting object-oriented design and construction of client-server applications and CASE integration.

HOTLINE: Why do you see object technology as a keystone for IS?

BRANSCOMB: Object technology can play an important role in streamlining existing CASE. There are already well over one billion dollars in investment in capturing business requirements in information models at major corporations. Object technology can help deliver on the promise of CASE by enhancing the existing tools with better downstream construction techniques and delivery architectures. The resulting improvements in productivity can pay back the investment. IDC did a study that indicated a 14:1 productivity advantage. We are more commonly seeing between 5:1 and 10:1, sometimes more.

HOTLINE: From the viewpoint of the IS community, what does Intellicorp bring to OT?

BRANSCOMB: The particular perspective that we bring is that we have significant practical field experience using O-O tools, applying them to complex, knowledge-oriented applications. We've been doing this since 1983 when we implemented KEE, the knowledge engineering environment. We can see the opportunity represented by object technology, but a lot of work is needed to make it real. From our experience we have learned what's good and powerful in OT. We've also learned that, to be accepted, OT needs to be packaged in a way that represents an evolutionary (not revolutionary) approach.

HOTLINE: That sounds like hard-won experience from the AI revolution.

BRANSCOMB: Yes. Compare the history of relational technology to that of AI and rule-based technology. One

Robert Shelton is Editor of this newsletter.

resulted in a huge industry; the other did not. Providers of relational products worked hard to bring relational to the customer, completely impeeded in and as an integrated step in what they already knew. Customers were not asked to throw away their existing technology. People need to see the technology revolution as something approachable. In general, AI did not succeed at this. For object technology to succeed, we must deliver true object technology products, but put the technology to use in a way that fits with the CASE platform strategy, people, and practices that mainstream IS is already accustomed to. This is no mean trick. We are asking IS people to embrace a better way to build software, but in the context of their existing infrastructure.

> ***It takes a very sophisticated hybrid architecture to take advantage of and enhance the legacy of existing application code.***

HOTLINE: This seems equally achievable with a pure object-oriented development environment capable of client-server integration with existing relational database management systems. What's new?

BRANSCOMB: Anyone can generate SQL and get at the data—that's not the point. It takes a very sophisticated hybrid architecture to take advantage of and enhance the legacy of existing application code. Pure object environments like Smalltalk can't do that. We have to extend and streamline the CASE process, and make more effective the process of embracing new platform economics. Object technology must support moving from a mainframe-centric view to a client-server or workstation LAN-based architecture. This implies a lot of things. Can the tools integrate with the transaction processing environments like IBM's CICS or AT&T's Tuxedo? Such capabilities hold great promise for improving return on investment on applications. How effectively can they take advantage of the economics of client-server architecture? Just being able to connect to an RDBMS is not the whole answer.

HOTLINE: Does the existing infrastructure into which you're integrating really work?

BRANSCOMB: No. The key is improving on what does work. A lot of companies can package object technology for software development, but few can deliver it as an improvement to the existing infrastructure. The problems of software development for individuals or small groups are not as severe as the problems at the corporate level. At the corporate level the problems are so severe they threaten the competitiveness of the company and the country. Object technology will have more impact at the corporate level.

HOTLINE: So you see IS as struggling to change and fix this corporate-level crisis?

BRANSCOMB: IS has been motivated to do this for many years. Many people have knocked IS because they were slow to adopt PCs. This is not a true perception. They were not against technology change. IS was concerned about security and sharing of data. This is the corporate crisis, and it's been seen as the traditional responsibility of IS. Like so many new technologies, the first commercial efforts in O-O come directly from the research lab. Many of the Smalltalk-based efforts are well intentioned, but they may be bringing too much of a research orientation to a marketplace that demands consistency, integration, imbedibility, and approachability.

HOTLINE: Intellicorp is taking what I would consider to be an uncommon approach for an object development tool vendor—providing a visual object front end and delivering programs in a standard, non-object language. You are evidently emphasizing object-oriented thinking instead of new languages with object-oriented syntax. Is there a message here?

BRANSCOMB: It's possible to give people the best of both worlds. We have tried to do this with PROKAPPA by generating ANSI C. If the object concepts are not working for a particular customer, the IS user can drop back to their old way of doing business—they can work in C. Smalltalk is autistic—it knows all about itself and only about itself. The corporate market generally is not interested in autistic tools. To get object technology into IS, we must provide something that even the most conservative or insecure IS person can feel comfortable using—taking a step forward to embrace the new way of doing business

HOTLINE: What is needed to make object technology a mainstream concept that wins IS acceptance?

BRANSCOMB: We see three things as critical in terms of making the object technology community into an industry. First, delivering a set of tools targeted toward getting the best out of the existing platforms. This means they need to be embedded into the existing database management systems, platforms, graphical user interface technology, and product sets. We have to allow people to use those parts of the old 3GL methods that are attractive, but eliminate the drudgery. Second, we have to deliver a methodology that is designed to be a seamless transition or approachable step from their existing information engineering methods. James Martin does have the confidence and commitment of the IS establishment. His critics may not feel that his methodologies are as elegant or innovative as some of the newer, more radical object-based approaches, but as he embraces OT, we have confidence he will do it in a way that IS will trust and accept.

HOTLINE: In other words, you can talk about the right way, but if people don't use it, it's worthless?

BRANSCOMB: Right, you must take them smoothly to the new way of doing business from where they are today. That's second. Now, the third factor is the development of a true industry community. That's where OMG plays an important role. IBM played that role for the RDBMS community by bringing together the right players around the SQL standard and relational technology. The OMG has made the right first step with the object request broker [Ed: ORB]. Now it is extremely important that the participants work together to achieve interoperability, a consistent and approachable methodology, and a set of marketing practices that encourage technical cooperation and technology sharing. OMG may be an appropriate industry forum for this cooperation. We need to ensure intense competition and concurrent intense cooperation. As an industry, we must share what we learn so that we push the field forward, but the customers need to get the benefits of competition.

HOTLINE: You see the OMG as a vital part of the future success of commercial object orientation?

BRANSCOMB: The OMG is an important organization. It is a forum for technical cooperation, encouraging interoperability, and allowing for more open competition in a way that serves the customers' interests because the products are interchangeable around the same APIs. The OMG is a forum for combining marketing muscle to make the whole pie bigger, and for sharing the benefits of the technology with the customer base at large. There's also a certain degree of the peer group keeping each other honest about what we're really bringing to the market.

HOTLINE: Intellicorp is teaming up with James Martin in its drive to win the IS community. Though his information engineering is well established in many of the more advanced IS shops, he's about the last person I would associate with object technology. Sounds like Ed Yourdon going object all over again. Why the teaming?

BRANSCOMB: James Martin has given credibility to IE concepts, and can do the same for OT from the IS perspective. The fact that he has reached out and taken an interest in a company that has always been considered technology rich may indicate that he sees the advantage of bringing radical technology into the IS world. Our experience bringing this technology to work is one of the things he is looking to benefit from by becoming a shareholder. [Ed note: Martin's open market purchase was 5.4% of Intellicorp common shares according to his SEC filing, with a maximum of 10%.] I think that the fact that someone as closely associated with the most conservative and traditional IS technology is embracing OT is an indication of the potential importance of the technology to IS, and the seriousness of the problems that this technology can address.

> ***Historically, software has not easily represented the goals, assertions, business policy, and procedures of our organizations.***

HOTLINE: So the underlying strategy is adherence to standards—existing platforms, environments, technology—while introducing change?

BRANSCOMB: I call it environment-centric architecture: Fitting into the existing systems and applications is as important as the purity and robustness of the technology implementation itself. Other vendors ask the IS customer to buy a new compiler—C++ or Smalltalk. That approach introduces a huge number of portability, complexity, and maintenance issues. We chose to take the extra effort to ensure that object technology-based applications could be implemented in a standard, SAA language, ANSI C. It's more likely that the existing programmer staff, tools, and applications in any given corporate IS shop will be in a standard language than a new one. What PROKAPPA delivers is the ability to use the staff, tools, and applications components while taking advantage of object orientation. We want to give IS a diamond in the setting of their choice.

HOTLINE: Because Intellicorp will be working with

≡ Your Turn ≡

How about it? How are we doing? This is the place to send us your message. Tell us what you like and what you don't, what you'd like to see but don't — or whatever other feedback you'd like to send. Send comments to Robert Shelton, 1850 Union Street, Suite 1584, San Fransisco, CA 94123; Fax: (415)928-3036; or, if you're ordering, send along with your order.

Please check whichever box applies:

	Read entirely	Scanned	Found helpful	Will refer to	Didn't read
Feature	❑	❑	❑	❑	❑
Engineering Methods	❑	❑	❑	❑	❑
Professional Development	❑	❑	❑	❑	❑
Software Documentation	❑	❑	❑	❑	❑
Product Announcements	❑	❑	❑	❑	❑
FYI	❑	❑	❑	❑	❑

Please rate the following ***1*** (dislike or strongly disagree) to ***10*** (like or strongly agree):

Presentation of material __ Comments: _____

Accuracy of material __ Comments: _____

Overall helpfulness in your job __ Comments: _____

What else would you like to see in THE HOTLINE? _____

Any other comments? _____

Name, title & phone (optional): _____

hotline on OBJECT-ORIENTED *technology*

❑ ***Yes,*** plug me into the latest thinking and developments in object-oriented technology. Enter me as a subscriber at the term marked below and rush me the current issue. This is a risk-free offer — I may cancel my subscription at any time and promptly receive a refund for the unused portion.

1 year (12 issues) ❑ $249

2 years (24 issues) ❑ $478 *(save $20)*

(outside US add $30 per year for air service)

Back issues @ $25 each *($27.50 foreign)*:

Vol.1, Nos. _____

Vol.2, Nos. _____ Vol.3, Nos. _____

❑ **Phone order**
Call Subscriber Services
(212)274-0640

❑ **Fax order**
Fill out term and name/address information, then fax to:
(212)274-0646

❑ **Send me a copy of** THE INTERNATIONAL OOP DIRECTORY
@ $73 ($81 foreign)
(all directory orders must be prepaid; fax or mail credit card information to SIGS, 588 Broadway, #604, NYC 10012; make check payable to OOP Directory. Foreign orders must be prepaid in US dollars drawn on US bank.)

❑ **Bill me**
❑ **Check enclosed**
❑ **Credit card orders**
❑ MasterCard
❑ Visa

Mail (and make out check, if applicable) to:
THE HOTLINE
Subscriber Services
P.O. Box 3000, Dept. HOT
Denville, NJ 07834
(foreign orders must be prepaid in US dollars drawn on a US bank)

Card# Expiration Date

Signature

Name

Company

Street/Building#

City/Province

ST/Zip/Country

Telephone

Send me a complimentary copy of your related publication(s):

❑ THE JOURNAL OF OBJECT-ORIENTED PROGRAMMING

❑ OBJECT MAGAZINE

❑ THE C++ REPORT

Call Subscriber Services
(212)274-0640
or send order to
SIGS, 588 Broadway, #604
New York, NY 10012

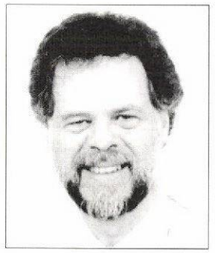
D2DC

hotline on OBJECT-ORIENTED *technology*

VOLUME 3, NUMBER 6 THE MANAGER'S SOURCE FOR TRENDS, ISSUES & STRATEGIES APRIL 1992

Thinking the unthinkable: reducing the risk of failure

To put it bluntly, using object-oriented technology (OOT) on a project can increase the probability that the project will fail.



Burton Leathers

Adoption of any innovative or unfamiliar technology entails an increased risk relative to the use of established, well-understood technologies. Even within the short history of computing, OOT must be considered an innovation, even a revolutionary innovation. Hence the adoption of OOT on a project increases the probability of failure.

Failure is not a word normally used in polite discussions of a technology about which most readers of this newsletter have positive, if not enthusiastic, feelings. Nonetheless, it is a critical consideration for anyone who wishes to try OOT. It is not a possibility that can simply be wished away. The possibility of failure in an OOT-based project is an issue that must be faced squarely. By thinking about failure, it is possible to reduce the probability that failure will occur and the costs incurred if it does occur.

SOME SIMPLE ACCOUNTING

In the commercial world, most software projects are undertaken because they are expected to yield a tangible economic benefit. The resulting software may be sold or rented, or it may be used to reduce costs in an existing operation. Whatever the mode in which the economic benefit is to be realized, there is a quantifiable benefit, which we may term the payback.

Against the payback must be set the cost of producing the software: the expenses for the project. The expenses serve to reduce the payback so that we may define:

Net economic benefit = Payback – Expenses

So far, so good, but the picture is not yet complete. In any project, there is always a risk of failure so it is necessary to discount the payback statistically in terms of the probability of success. The project benefit is then restated as:

Expected net economic benefit =
Pr(Success) × Payback – Expenses

Expressing the probability of success in terms of the probability of failure and rearranging terms yields the interesting expression:

continued on page 4 ...

IN THIS ISSUE

FEATURE — Thinking the unthinkable: reducing the risk of failure *Burton Leathers*	1	2	FROM THE EDITOR
ENGINEERING METHODS — Mitigating madness with method: first establish what you value *Michael Fuller*	7	20	PRODUCT ANNOUNCEMENTS
PROFESSIONAL DEVELOPMENT— Championing object technology for career success in the 1990s *K.L. Straat*	11	22	NEW PARTNERINGS & ACQUISITIONS
SOFTWARE DOCUMENTATION— Objects and actions in end-user documentation *Tony Durham*	14	23	CALENDAR
FYI — Industry publication excerpts	18	24	YOUR TURN

From the Editor



We spoke last month of 1991 as the opening act in the commercialization of object technology. Moving object technology from modest projects and research efforts to enterprise-sized systems and turnkey consumer products, though, is a challenge that places responsibilities on this fledgling industry that many technology providers are ill equipped to handle. Unfortunately, commercial software vendors and information technologists can take little comfort from the performance of most of their predecessors. With each new wave of "inherently superior paradigm," vendors, technology users, and end-product users have struggled with basic issues that still beg for solutions: How do I successfully introduce this new technology into my organization? How can we reduce quality deviation and achieve real reuse of tested, proven software components? What should we deliver to our user-customers in the way of documentation? And, by the way, what opportunities and risks does this new technology hold in store for my own personal development? These are the issues that this month's HOTLINE authors address.

From the viewpoints of our columnists, one can also find direction for beginning to address some of the demands that business managers will place on the object technology marketplace over the next year:

- quantify and document the reuse available with objects and show me analysis and design methods that will help our shop achieve those results
- demonstrate the real return on my investment in information technology and quantify the cost reduction that can be achieved using object technology
- show me solutions for sharing business information among hundreds or thousands of concurrent users that will help me develop applications for our marketing, product development, manufacturing, and customer-service users—before they miss the market window
- give me a tool set that will get our software to market within our business time constraints that also addresses the configuration management issues of large-scale deployment and software maintenance
- prove that interoperability is cost-effective, and show me how to achieve it with the mix of technology we have already purchased.

These, I think, are real and pressing demands that vendors and technologists staking their claim with object technology cannot afford not to answer. Today, our answers are generalizations and vague representations—not hard facts. These answers will not appear of themselves. What we can show for this technology is a function of how we approach it and how effectively we apply it.

In this month's feature, **Dr. Burton Leathers** of COGNOS addresses an issue fundamental to change, growth, and new technology: failure. I posit that learning and growth arise primarily from failure, and less so from success, because we as humans are more imitative and less introspective about our successes. Most of us try to improve on our failures—provided that we have not been conditioned to run from them in denial. Consider, in contrast, that *kaisen*, a Japanese business term for continuous process improvement, involves systematically learning from success as well as failure—a competitive advantage that more than doubles the availability of learning opportunities in any organization that succeeds more often than it fails.

Yet, Dr. Leathers observes that the cost of failure tends to be disproportionate to the cost of success, a phenomenon largely caused by panic and denial behaviors that inhibit constructive learning, rational decision making, and clear perception of the facts of one's circumstances.

His postulate rings true with my own experience. The largest software project failure I've been party to was, ironically, one of the first large projects of my career. My employer sunk over $20 million (US) into the effort over two years, with zero return on the investment. Originally, that project had been budgeted at a few million dollars. The frustration and disappointment left most of the project team members depressed and underproductive for months afterward. In the user community, the consequent mistrust of the company's information technology staff plagued the organization for years and is still evident today. In the project autopsy, a dollar value was never assessed on these hidden hu-

Editor
Robert Shelton

SIGS Publications, Inc. Advisory Board
Tom Atwood, *Object Design*
Grady Booch, *Rational*
George Bosworth, *Digitalk*
Brad Cox, *Information Age Consulting*
Chuck Duff, *The Whitewater Group*
Adele Goldberg, *ParcPlace Systems*
R. Jordan Kreindler, *General Electric*
Meilir Page-Jones, *Wayland Systems*
Tom Love, *Consultant*
Bertrand Meyer, *Interactive Software Engineering*
Sesha Pratap, *CenterLine Software*
P. Michael Seashols, *Versant Object Technology*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

HOTLINE Editorial Board
Jim Anderson, *Digitalk, Inc.*
Larry Constantine, *Consultant*
Mary E.S. Loomis, *Versant Object Technology*
Reed Phillips, *Knowledge Systems Corp.*
Trygve Reenskaug, *Taskon A/S*
Zack Urlocker, *Borland International*
Steven Weiss, *Wayland Systems*

SIGS Publications, Inc.
Richard P. Friedman, Founder & Group Publisher

Art/Production
Kristina Joukhadar, Managing Editor
Pilgrim Road, Ltd., Creative Direction
Elizabeth A. Upp, Production Editor
Caren Polner, Desktop Designer

Circulation
Diane Badway, Circulation Business Manager
Kathleen Canning, Fulfillment Manager
Vicki Monck, Circulation Assistant
John Schreiber, Circulation Assistant

Administration
David Chatterpaul, Bookkeeper
Frank Ferrantelli, Bookkeeper
Claire Johnston, Conference Manager
Cindy Roppel, Conference Coordinator
Jennifer Fischer, Public Relations
Jennifer Englander, Administrative Assistant

Margherita R. Monck, General Manager

Jane M. Grau, Contributing Editor

hotline on OBJECT-ORIENTED *technology*

THE HOTLINE ON OBJECT-ORIENTED TECHNOLOGY (ISSN #1044-4319) is published monthly by SIGS Publications, Inc., 588 Broadway, NY, NY 10012, (212)274-0640. © Copyright 1992 SIGS Publications, Inc. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the U.S. Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher. Mailed First Class. Subscription rate — one year (12 issues) $249, Foreign and Canada $279. Single copy $25.

POSTMASTER: Send address changes & subscription orders to The HOTLINE, Subscriber Services, P.O. Box 3000, Dept HOT, Denville, NJ 07834.

Submit editorial correspondence to Robert Shelton, 1850 Union Street, Suite 1548, San Francisco, CA 94123. Voice: (415) 928-5842; fax: (415) 928-3036.

SIGS PUBLICATIONS

Publishers of *Hotline on Object-Oriented Technology, Journal of Object-Oriented Programming, Object Magazine, The X Journal, The C++ Report, The Smalltalk Report,* and *The International OOP Directory.*

hotline on OBJECT-ORIENTED *technology*

Back issues available

Vol..3, No.5/March '92 ≡ A large-scale users' assessment of object orientation • *Plant* ≡ Report on the Object-Oriented COBOL Task Group • *Adams* ≡ Interview with K.C.Branscomb • *Shelton*

Vol..3, No.4/February '92 ≡ The big prize: acceptance of O-O by the MIS community • *O'Shea* ≡ Retrospective: 1991—The year it all changed • *Tom Love* ≡ Making the transition to O-O technology • Ron Suarez, PhD ≡ Interview with Beatriz Infante • *Shelton*

Vol. 3, No. 3/January '92 ≡ Enterprise object modeling: knowing what we know • *Shelton* ≡ Adopting objects: pitfalls • *Connel* ≡ Adoption rate of object technology: a survey of NSW industry • *Henderson-Sellers.*

Vol..3, No. 2/December '91 ≡ Accepting object Technology • *Bennett* ≡ Adopting objects: a path • *Connel* ≡ Incorporating graphical content into multimedia presentations • *Berman & Weiss*

Vol.3, No. 1/November '91 ≡ Leading the U.S. semiconductor manufacturing industry toward an object-oriented technology standard • *Hollowell* ≡ Coping with complexity: OOPS and the economists' critique of central planning • *Lavoie, Baetjer, & Tidloh* ≡ Choosing Object Technology: What's the object? • Rosenthal ≡ OOP: the MISsing link • *Rovira*

Vol.2, No. 12/October '91 ≡ A modest survey of OOD approaches • *Budman* ≡ What is a "certified" object programmer? • *Bellin* ≡ Perspective: investing in objects today •Bowles ≡ Object oriented in Melbourne, Australia • *Haebich* ≡ The Object Management Group • *Gutman & Matthews*

Vol.2, No. 11/September '91 ≡ From applications to frameworks • *Urlocker* ≡ Report on the Object-Oriented COBOL Task Group • *Adams & Lenkov* ≡ Getting started with object technology: efffectively planning for change • *Lorenz* ≡ Object statistics on the way • *Lenzi* ≡ On objects and bullets • *Onart*

Vol.2, No. 10/August '91 ≡ Distributed object management: improving worker productivity• *Osher* ≡ Getting the best from objects: the experience of HP • *Coleman & Hayes* ≡ APPLICATIONS: EC employs object technology … • *Stein* ≡ CAPACITY PLANNING: Fiddling while ROMs burn • *Rovira*

Vol.2, No. 9/July '91 ≡ Multimedia is everywhere! • *Weiss & Berman* ≡ Developing an object technology prototype • *Kulinek* ≡ Object-oriented capacity planning • *Rovira* ≡ How OOP has changed our developmental lifecycle • *Zeik* ≡ Modularization of the computer system • *Angenius*

Vol.2, No. 8/June '91 ≡ Domain of objects: the Object Request Broker • *Dyson* ≡ Object-based approach to user documentation • *Reeder* ≡ Report on rhe Object-Oriented COBOL Task Group • *Adams & Lenkou* ≡ Do we need object-oriented design metrics? • *Hopkins*

Vol.2, No.7/May '91 ≡ Hybrid object-oriented/functional decomposition for software engineering • *Henderson-Sellers* ≡ So, what makes object databases different? (Part 4) • *Blakey* ≡ Using the generic application to solve similar domain problems • *Gossain* ≡ Experiences using CLOS • *Hopkins* ≡ International Conference on Object-Oriented Technology, Singapore • *Bennett*

Vol.2, No.6/Apr. '91 ≡ An artist's perspective of programming • *Rovira* ≡ So, what makes object databases different? (Part 3) • *Blakey* ≡ Moving from Pascal to C++, Part 3 • *Gole* ≡ Object projects: what can go wrong • *Stewart* ≡ Reflections from LOOK-'91 • *Anders-Argenius*

Vol.2, No.5/Mar. '91 ≡ Large-scale object systems development • *Morgan & Davis* ≡ Data definition languages • *Blakey* ≡ Object disorientation • *Kerr* ≡ Moving from Pascal to C++, Part 2 • *Keller* ≡ The Object Management Group • *Barber*

Vol.2, No.4/Feb. '91 ≡ Moving from Pascal to C++ • *Sollisch* ≡ The Australian object-oriented scene • *Henderson Sellers* ≡ So, what makes object databases different? • *Blakey* ≡ Managing the evolution • *Novobilski* ≡ 1990: the year of realism • *Love* ≡ Choosing a configuration for object development • *Moody*

Vol.2, No.3/Jan. '91 SPECIAL ISSUE: CASE AND METHODOLOGY ≡ Object orientation: stop, look and listen! • *Page-Jones* ≡ The spiral model for object software development • *Wasserman & Pircher* ≡ Bootstrapping object-oriented CASE • *Wybolt* ≡ A CASE tool for object software development • *Ackroyd* ≡ Comments on "Race for the database pennant" • *Loomis*

Vol.2, No.2/Dec. '90 ≡ Waiting for OODB2 • *Durham* ≡ Object modeling • *Singh* ≡ Object-oriented COBOL — the next generation • *Adams & Lenkov* ≡ Object-oriented training in the UK • *Hopkins*

Vol.2, No.1/Nov. '90 ≡ Objects by teamwork • *Constantine* ≡ Race for the database pennant: a third-generation manifesto • *Stein* ≡ Project team descriptions *Goldberg & Rubin* ≡ The Australian object-oriented scene • *Henderson-Sellers*

Vol.1, No.12/Oct. '90 ≡ Crown jewels: the value of OOT in MIS departments • *Bonar* ≡ Why is an object database faster than a relational system? • *Loomis* ≡ Visual programming object language environments • *Klausner* ≡ Hype — necessarily (or a necessary) evil? • *Adams* ≡ ON MANAGING: Déjà Vu • *Lenzi*

To order, see back page; for reprints of individual articles, call Michael Biggerstaff at Reprint Management Services (717) 560-2001.

Calendar

THE HOTLINE CALENDAR presents conferences and meetings that focus exclusively on object-oriented technology. To have a meeting or conference listed, please send the dates, conference name and location, sponsor(s) and contact name and telephone number to the Editor: Robert Shelton, 1850 Union Street, Suite 1584, San Francisco, CA 94123; fax: (415) 928-3036.

April 27–30	June 1–5, 1992	June 16–18, 1992	July 14–17, 1992	Nov. 13–16, 1992
XWorld	**Object Expo**	**Software Development '92/ Wembley Exhibtion**	**Object Expo Europe**	**C++ World**
New York, NY	New York City	Europe	London	Secaucus, NJ
Sponsor: THE X JOURNAL SIGS Conferences	Sponsor: JOOP, OBJECT MAGAZINE SIGS Conferences	Sponsor: Blenheim Online	Sponsor: C++ REPORT OBJECT MAGAZINE, JOOP, COMPUTING, SIGS Conferences	Sponsor: C++ REPORT SIGS Conferences
Contact: Claire Johnston (212) 274-0640	Contact: Claire Johnston (212) 274-0640	Contact: (081) 742-2828	Contact: Claire Johnston (212) 274-0640	Contact: Claire Johnston (212) 274-0640

New Partnerings & Aquisitions, April 1992

Servio announced that Lacy Edwards, former president and CEO of XA Systems, has been appointed to the position of President and CEO. Edwards replaces Donal O'Shea, who has left the company to pursue other interests.

Digitalk, Inc. announced the acquisition of **Instantiations, Inc**. Instantiations, a Smalltalk training, consulting, and product development company, provides a wide range of services to Fortune 500 companies that are developing applications utilizing Smalltalk object-oriented technology.

ONTOS established several new sales offices and hired new sales and marketing directors. ONTOS has hired Aileen Black, formerly the national director of federal accounts for Oracle Corp., as director of ONTOS' Federal Group and Southeast region, based in Vienna, VA. Lynn Currier, most recently with Bachman Information Systems as Northeast regional sales manager, is responsible for the Northeastern region and Eastern Canada out of ONTOS' headquarters in Burlington, MA. Steve Tonissen, formerly with Pansophic Systems as director of CASE, has set up ONTOS' Midwest office located in Chicago. Deborah Kelfer, formerly manager of worldwide marketing at Xerox Corp., now heads ONTOS' Western region office based in Los Angeles. Sherry Collins joins ONTOS as marketing director, responsible for strategic marketing, product positioning, marketing communications, and market research. Sherry was most recently president of Clarity Associates, where she guided Bachman Information Systems' marketing efforts in preparation for its initial public offering.

Objectivity Inc. announced that it has reorganized its senior management team and support organizations. Atam Lalchandani was promoted to President and Chief Executive Officer, effective immediately. Lalchandani was previously Vice President of International Operations and Chief Financial Officer. The action will reorganize Objectivity into four separate groups: product operations, North American sales and marketing, international sales and marketing, and finance and administration. The former technical services and engineering groups were merged to form a single organization responsible for engineering, customer support, and product management. Ed Thompson's new responsibility will be to head the new product operations group as Executive Vice President. Prior to joining Objectivity, Thompson was General Manager of the advanced products division at Mentor Graphics. Jerry Deems was promoted to Vice President, sales and marketing, North America. Deems was previously Director of Sales, North America. Drew Wade, Vice President, will continue to represent Objectivity within industry standards organizations and will assume responsibilities for product management. Bob Field, former CEO of Objectivity, will continue to serve on the board of directors and assist the company as a consultant.

The **French Geographical Survey** (BRGM) and **Societe des Outils du Logiciel** (SOL), one of the leading European developers of the object-oriented language Eiffel, signed a long-term joint technology contract for object-oriented graphical applications, CASE tool development, and training and consulting. BRGM, the **Bureau de Recherches Geologiques et Minieres,** intends to switch to Eiffel as the development platform for new large-scale projects, in particular in the area of Geographical Information Systems (GIS). BRGM is an internationally recognized federal French agency. It conducts research on petroleum and water reserves as well as on environmental pollution for both public and private customers, many of them oil companies.

Object Design announced a licensing agreement under which **IBM Corporation's** Technology Products group will use Object Design's ObjectStore object-oriented database management system in its electronic design automation (EDA) systems. IBM's Technology Products group will employ ObjectStore to build RISC System/6000-based electronic design framework applications, as well as a suite of integrated applications for use by IBM's designers.The Technology Products group will use ObjectStore to develop EDA software applications to be used for storing and managing its design data for EDA. As part of the agreement, ObjectDesign also will provide the group with onsite technical support.

NCR Corporation and **Lucid, Inc**. announced joint development to advance technology for future software application development environment products for the NCR System 3000 family of Intel-based general-purpose computers. The Lucid technology is based on a client/server architecture and provides an integrated environment for developing and understanding C and C++ programs.

Object Technology International, Inc. announced a new development agreement with IBM, Inc. The new agreement with IBM's Applications Business Systems (ABS) will enhance the use of graphical user interfaces and objects in application development. ABS will work with Ottawa-based OTI to create an object-oriented environment for developing cooperative applications. OTI will combine the object-oriented ENVY/Developer technology with AS/400 cooperative processing support to provide a client-server application development environment for Smalltalk/V programmers.

Liant Software Corporation acquired C++/Views, a C++ class library and source code browser, from **CNS, Inc.** Financial terms of the acquisition were not disclosed.

man costs. Conversely, the largest financial success from a software project I've worked on mitigated a $650 million environmental liability at a cost (to the same employer) of about $5 million. As is often the case among us humans, though this success has not been forgotten—especially at the executive level—far fewer people know the "numbers" on this project than bitterly recall the exact dimensions of the earlier failure. As Dr. Leathers points out, managing the cost of failure can prevent an unsuccessful project from becoming a disaster.

The ability of players in an organization to manage the risk and consequences of failure is a function of the organization's culture and business management processes that this culture reinforces. Dr. Leathers offers hard-won insights into processes for introducing technology change that permit management of failure's cost while maximizing the learning value of either success or failure. Those of us who use object technology today have an opportunity to apply these insights from the start. For some of us, unfortunately, the impetus to change our approach to technology and project management will not come until the pain of failure is overwhelming. Facing such failure as a defender of technology radically different from today's mainstream tempts a backlash—outright rejection of this and other new computing technology—and a resulting long-term loss of opportunity. Therein lies another substantial hidden cost to uncontrolled failure that should give us early implementors food for thought and a reason to take Dr. Leathers' advice to heart.

Fundamental to successful product delivery—and also an effective failure avoidance strategy—is the concept of kaisen: the continuous improvement of process. Understand the causes of failure and success; repair and improve the process that creates the result. Historically, attempts to fortify the software development process have met limited success. Those who have observed the implementation of engineering disciplines in multiple organizations see a pattern: belief in the superficial technology fix in lieu of real change to the underlying development processes. This superficial focus masks an underlying lack of shared values, and the lack of shared values leads to failure.

In our last issue, **Mr. Michael Fuller** reviewed the first of two books by Sally Schlaer and Stephen Mellor on their proposed method for object-oriented software development. He joins us in this issue, taking a back-to-basics approach to software engineering methods and discussing his thoughts on finding a workable method. Without first addressing what we value, Mr. Fuller observes, we cannot agree on a suitable process for achieving a common, desired result.

Mr. Fuller's article could not be more timely, as the issue of engineering methods for object development is becoming unavoidable. We're using object technology on real projects now. Real projects involve real money, real gains, and real loss. Already, this industry has seen several bet-the-company projects. Some, such as Mentor Graphics, have lost the bet and survived to try again; Others, like Momenta, succeeded the first time, and now must prove the durability of their process through the product lifecycle. Even in the absence of well-defined engineering disciplines for object development, shared values—especially a very clear understanding of exactly what the project must offer the business to make a profit and deliver a desirable product—are a major determinant of outcome. Not surprisingly, shared values are as critical to the selection of development methods as to strengthening our business's non-software product delivery processes. Are we seeing, perhaps, the first influx of nuts-and-bolts business reality into software development? I hope so!

Business evaluates technology in terms of return on investment and opportunity cost—what did we get for the money, and what could we have had if we had spent the same money elsewhere or not at all? Opportunity cost is not only financial. Missing a market window or being inadequately competitive has a cost that can be estimated, but not measured in absolute dollars. The cost of being second is the harsh reality of having your product defined by the competition instead of being the market-defining product, the cost of having to convince your prospective customers that your product is something more than a "me too." The cost of being second can be the greater portion of a marketing and sales budget that produces weak margins.

Mr. Tony Durham, our European Correspondent, looks at one way successful competitors will differentiate themselves and their software products in the future: product documentation. Sure, documentation is obligatory today, but how often is it more useful than frustrating? How much are we missing from today's software products because we don't know what's there? As software products increasingly involve the user directly with software objects, a different perspective is made available. Documentation can focus on the capabilities and characteristics of the objects themselves instead of centering on fixed work tasks.

Mr. Durham's concept of software products as tool kits instead of monolithic, narrow-purpose solutions suggests one of the great potential gains from the interoperability standards being developed by the Object Management Group (OMG): end-users could eventually plug-and-play tools from different products. Standards are an enabling factor. Mr. Durham's proposal for object-oriented documentation is clearly an enabler for the user. End-users cannot learn and effectively use tools that are documented with the expectation that everyone will use the software in the same, program-oriented way. Turning over control of the components to the user may significantly change the way we regard and use software.

Finally, **Mr. Kent Straat** of Object Resources International brings us a perspective on managing personal success through the tumult and uncertainty that many organizations are experiencing today. Many of the themes that recur as fundamental to a successful implementation of object technology and failure risk minimization are equally relevant to a successful career. Mr. Straat, as a professional career coach and recruiter, suggests how these may be used to advantage—how to ride out some of the uncertainty that is an unavoidable (and I think potently valuable) aspect of technology-driven change.

As the expression goes, this is no longer sandlot touch football! With the beginning of serious commercialization, object technology and the companies that provide it are having to face business to win business. It's our opportunity to prove our value.



[***Editor's note:*** *I would like to acknowledge the work of J.M. Edwards, who coauthored the Objects Down Under column with B. Henderson-Sellers in the January issue of* HOTLINE. *His name did not appear in the byline for that column and apologies are extended to him for the oversight.*]

...continued from page 1

Expected net economic benefit =
Payback – Expenses – Pr(Failure) × Payback

It is true that a given project succeeds or fails and so it either realizes the entire net economic benefit or is out the amount of the expenses. The point is a set of projects will, on average, yield the sum of the expected net economic benefits.

Unfortunately, this is as yet not quite the full picture. In far too many instances, failure is not a very clean process. It is often the case that when a project does fail, the expenses tend to exceed those that would have occurred had the project succeeded. Because very few organizations plan with the possibility of failure in mind, failures tend to be unduly messy and correspondingly costly.

To complete the simple accounting picture it is also necessary to add the effect of the cost of failure. The full picture then becomes:

Expected net economic benefit = Payback – Expenses – Pr(Failure) × [Payback + Cost of Failure]

It can easily be seen that for a project with fixed payback and expenses, the key to maximizing the expected net economic benefit is to minimize both the probability of failure and the cost of failure.

THE NATURE OF FAILURE COST

One reason the costs of failure as opposed to project expenditures receive little attention is because they rarely appear in direct association with the failing project. They generally have three distinguishing characteristics: They are indirect costs, frequently nonmonetary, and generally result from failures of organization of the project rather than from a failure of the technology itself.

One of the most curious costs of failure—the devaluation of existing technology and work on existing products—is a cost often associated with successful projects as well, where it is masked by the success of the project. One of the strategies for adopting a new technology is to denigrate existing technology and the work on products based on it. If the new technology is successful, the devaluation of the existing technology and products is a marginal problem as they are soon obsolete. However, if the new technology fails to supplant the old, all that is left is a failed technology and poorly supported products based on a devalued technology.

Related to this is the natural tendency to presell the new technology to the existing customer base, especially to an internal customer base. When the new technology fails, the customer base is left confused and suspicious.

Because the project that fails is generating some sort of product, the failure means that the market window is missed and revenue (savings) is deferred or lost.

Project failure also has strong negative impacts within the development organization. The simple fact of failure means a drop in morale and—in the limiting case—a loss of personnel for whom the new technology is or has become critical.

Finally, if the management of the failure has been weak, it is highly likely that a panic response has set in as last valiant attempts are made to salvage the project. Panic responses almost invariably result in expenses beyond those originally planned.

It may be seen that many of the failure costs cited above are not the sort that get attached to project budgets. These real costs of failure include reduced support and delayed development for existing revenue-producing products, deferred or lost revenue, increased recruiting and training costs, and lowered morale and productivity.

BASIS FOR FAILURE

There are many different ways in which a technology adoption project can fail. It is critical to the effective management of potential failure to understand the main sources of project failure involving new technology adoption. The various specific failures can be grouped into four general types: technology deficiencies, technology-problem mismatch, technology-organization mismatch, and organizational deficiencies.

Technology deficiencies

The first suspicion about the failure of OOT adoption projects is that the primary cause of failure is the technology itself. In some instances this is true. A further suspicion is that the technology failures are due to the languages because, after all, O-O languages are novel. Sometimes this, too, is true. But blaming the technology is simply a matter of finding the easiest scapegoat—one that cannot argue back. Besides, if it is not the technology that is at fault, it is probably ourselves who are—and admitting our own faults is always much harder than blaming an external, inanimate object.

Still, it must be acknowledged that OOT suffers in some degree from uneven maturity. The deficiencies usually mentioned pertain to the languages themselves, but this simply indicates a focus on programming rather than process. Even with the most perfect of languages, a project can fail due to lack of good analysis and design or source and configuration-management tools.

In evaluating the specific technology to be employed, it is important to be sensitive to the relative strengths and weaknesses of available OOT. Even so, the best *a priori* evaluation can fail to identify a potential trouble spot simply because it is often difficult to know in advance to what characteristics of a technology a problem

The Whitewater Group
1800 Ridge Ave.
Evanston, IL 60201
708.328.3800

Whitewater introduced **ObjectGraphics** for C++, a high-level, object-oriented graphics library that extends the power of Borland C++ with application frameworks and Turbo C++ for Windows to create comprehensive, platform-independent support for graphics, including diverse shapes, bitmaps, icons, and multiple fonts and text styles in a Microsoft Windows application. ObjectGraphics for C++ completely masks the graphics engine in Windows and allows users to develop graphical applications using a simple set of graphic objects rather than function calls.

Expersoft
6620 Mesa Ridge Rd., Ste. 100
San Diego, CA 92121
619.546.4100

Expersoft's primary product line is **XShell**, an application framework that performs two basic functions. It provides object-oriented interface services that seamlessly and transparently connect multiple users with applications, databases, data input devices, and storage devices. It also performs difficult system functions through its intersection at the operating-system level to control execution across distributed sequential and parallel-processing platforms. XShell 2.0 is customized for software development of object-oriented software systems in distributed processing environments. XShell is a set of compilers, drivers, and software tools that supports and in some cases automates, i.e., generates the code for, building object-oriented software systems. XShell toolkits are also offered as separate products. The product line includes toolkits for building systems using rule-based inferencing (XRule), fuzzy associative memory (XFuzzy), genetic-optimization algorithm (XGenetic), neural networks (XNet), cartographic reasoning (XTerra), and hypermedia (XMedia).

Glockenspiel Limited
39 Lower Dominick St.
Dublin 1, Ireland

Glockenspiel is publishing their first open C++ toolset for the new IBM AIX Software Development Environment (SDE) WorkBench/6000. **Glockenspiel C++ 3.0** comprises an ANSI-conformant C++ compiler based on AT&T/USL C++ 3.0, a portable, ANSI-conformant preprocessor, and an open, user-extensible, table-driven driver program. These and other tools in the set (constructor linker, debug/browse support, and parameterized types repository manager) will cooperate with each other and non-Glockenspiel tools like C compilers, linkers, and editors via the IBM AIX SDE WorkBench/6000.

GUI Computer Inc.
PO Box 795908
Dallas, TX 75379
214.250.3472

GUI Computer introduced **ObjectTable C++**, a Windows-oriented library that implements a programmable, multicolumn table object. The table supports auto data validation for date, time, currency, and scientific notation data formats/ cut and paste to the Windows clipboard; capacity greater than 32K; and both horizontal and vertical scrolling. Incorporating a table object into an application is straightforward: the programmer defines table attributes (size, number, and widths of columns; column titles; validations, etc.) and then provides code to load data into the table or save it out as needed. ObjectTable C++ does all its own memory and event management internally.

TurboPower Software
PO Box 49009
Colorado Springs, CO
80949-9009
719.260.6641

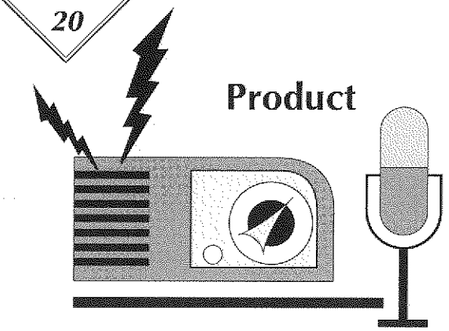
TurboPower Software announced **Object Professional for C++**, a library of test-mode user interface objects. Object Professional for C++ (OPC) is a straight port of the user interface objects from Object Professional for Turbo Pascal. OPC includes high-level objects such as text editors, dialog boxes, scrolling data entry screens, help systems, and more. The user interfaces are ready-to-use and built-in calls allow programmers to customize the behavior. OPC does not use event-driven programming. Applications built using OPC will run nicely in 640K 8088-based PCs.

ParcPlace Systems
1550 Plymouth St.
Mountain View, CA 64043
415.691.6700

ParcPlace Systems, Inc. announced **Objectworks/C++ Release 3**, a new release of its integrated C++ UNIX programming environment,which will also be available for IBM's RISC System/6000 product line. Previously, Objectworks/C++ was available for the Sun-3 and SPARCstation. The product now supports the standard Motif user interface, giving it an entirely new look and feel and replacing a proprietary interface. The Motif window manager now controls all Objectworks' windows, enabling programmers to work with all Motif productivity tools in a familiar way. Objectworks/C++ is now easier to use for programmers already familiar with the Motif interface, and will also provide extensive drag and drop facilities for choosing how to view program information, making the product more intuitive to learn and use. New features include an architecture to support AT&T's Language System Release 3.0 and performance advancements to support large system development. The programming environment now offers faster import and turn-around times compared to the earlier version.

Otter Research Ltd.
PO Box 265 Station A
Nanaimo, BC V9R 5K9
Canada
604.756.0956

Otter Research, Ltd. is shipping **AUTODIF**, a C++ class library combining array language extensions with the reverse mode of automatic differentiation for use in nonlinear modeling and statistics. Derivatives are calculated with the same accuracy achieved by analytical formulas and with minimal computational overhead. Existing C language computer code can be used with few changes. Extensive array manipulation features make it possible to easily and rapidly develop new applications. The-125 page user's manual includes sample applications in nonlinear parameter estimation, root finding, and neural networks.



Product Announcements

Product Announcements is a service to the readers of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY; it is neither a recommendation nor an endorsement of any product discussed.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
206.882.8080

Microsoft Corporation introduced **Microsoft C/C++ Version 7.0 Development System for Windows**, incorporating High-performance Object Technology (HOT) for the development of small, fast, Windows-based applications. The product contains a comprehensive set of professional tools for Microsoft Windows 3.1 so developers can immediately begin building applications for Windows 3.1 when it begins shipping, as well as for Windows 3.0 including the Windows debug kernel and set-up toolkit, plus a new version of CodeView for C++ debugging, a much faster Programmer's Workbench, and the Qualitas 386MAX memory manager to ease memory constraints during development. Microsoft HOT is a set of integrated technologies providing advanced C++ design, new compiler optimizations, and a Windows-based applications framework to maximize the performance of objects under Windows. Microsoft Foundation Classes provide objects for Windows with more than 60 C++ classes that abstract the functionality of the Windows application programming interface.

AICorp, Inc.
138 Technology Dr.
Waltham, MA 02254-9748
617.891.6500

AICorp, Inc. announced **INTELLECT/PC with Cooperative Processing**, a new version of its natural language information system that gives users conversational English access to complex databases, eliminating the need for users to learn command languages, syntax, or database structure. Primary applications include human resources and marketing, applications that are typically information intensive and difficult to access. The product adds three key features: a graphical user front end, interfaces to databases on the PC, and the connectivity option to automatically query information from either the PC, mainframe, or both. AICorp's Cooperative Processing facility provides the communication link between INTELLECT/PC and INTELLECT/mainframe, allowing INTELLECT to answer questions that require data stored on the mainframe or local PC or both. It also enables INTELLECT applications to be efficiently distributed over multiple platforms. INTELLECT/PC runs on both the OS/2 and DOS platforms. Included with the product are automatic database interfaces to OS/2 Database Manager, sequential files, and KDB (AICorp's knowledge database).

Code Farms Inc.
Richmond, Ontario
Canada
613.838.4829

Code Farms Inc. announced a new disk-saving utility program for their C and C++ data structure libraries. The algorithm, 20X faster than current utilities, was developed jointly by Code Farms and Zycad Corporation of Menlo Park, CA. The new software utility program allocates new objects from a set of memory pages that keeps a record of where all pointers are located, with an overhead of only 3.2%. The pages of memory, typically 1MB in size, are written to disk as binary blocks. When restoring the data to disk, only one index operation and one offset are needed to restore the pointers.

Hewlett-Packard
3404 East Harmony Rd.
Fort Collins, CO
80525-9599
303.229.4421

Hewlett-Packard introduced HP **InterViews Plus** software, a library of reusable components for accelerating the development of OSF/Motif p\graphical user interfaces. The object-oriented class library consists of almost 150 pretested classes of interactive object such as push buttons, dialog boxes and scroll bars. Developers can use these objects to construct custom user interfaces much faster than by using just the OSF/Motif tool kit. HP InterViews Plus is the first class library from HP for developing OSF/Motif applications using the C++ programming language. It is based on the INTERactive VIEWS (InterViews) GUI class library developed as part of a research project by Stanford University and Silicon Graphics Corp. HP added full support for OSF/Motif 1.1 and complete documentation, support, and training to HP InterViews Plus.

Inmark Development Corp.
2065 Landings Dr.
Mountain View, CA
415.691.9000

Inmark Development Corporation began volume shipping of its flagship product, **zApp**, an object-oriented applications framework for Microsoft Windows. zApp simplifies the creation of complex Windows applications by encapsulating the entire Windows API into versatile C++ objects and facilitating portability to other operating platforms. It features product documentation and technical support.

Send Product Announcements to Robert Shelton, 1850 Union Street, Suite 1548, San Francisco, CA 94123, fax: (415) 928-3036. Include company name, address, and phone number.

will be sensitive. A precise characterization of the critical areas of a problem in terms of traditional technology is a poor guide to potential problem areas with OOT.

The most disconcerting aspect of technology failure is that it can strike even those who are most conscientious in their evaluation of the technology. The technology may not have scaled or the *a priori* weighting of selection factors may not have reflected the reality of the technology in use. Many surprises await even the most careful innovator.

Technology-problem mismatch

Some people are all too happy to state outright that OOT is a niche technology and while there are some application for which it is suited, e.g., construction of GUIs, there are many others such as numerical analysis packages for which it is quite unsuited. There is a germ of truth in this. However, OOT proponents are quite clear that OOT offers very specific types of advantages and that only projects and application areas in which these can be exploited will manifest the expected benefits.

Jim Moore of IBM provides an interesting example of an application area in which one of the primary virtues of OOT reusability cannot be realized. Avionic software is a critical component of any modern fighter aircraft. Given the staggering volume of code required for new aircraft, it would seem a good place to enjoy the benefits of OOT. However, new fighters are turned out relatively infrequently and---even within a single country—the work is rotated amongst available contractors. As a result, the opportunity for reuse occurs only every decade, or even less frequently. In a sense, there is a problem-technology mismatch between OOT and avionics.

The key point is that OOT does not address all problems and not all software has as its primary shortcomings the areas which OOT best addresses.

Technology-organization mismatch

Some time ago I had a very interesting conversation with a senior manager in a large computer company. The people in his group had developed an O-O based analysis and design methodology for use in software development within his division. The development managers liked it; the developers liked it; the accountants didn't like it because it replaced a traditional phase-review waterfall technique with a rapid prototyping, spiral technique. The accountants didn't know how to release project funds other than in large dollops at well-defined milestones. This very good methodology, wanted by the engineering organization, failed because it ran too strongly against the prevailing organizational culture.

Unfortunately, it is possible to cite many other ways in which the prevailing organizational culture can clash with effective utilization of OOT. The critical factor is that OOT is not simply a new way of writing code, it is a new way of engineering software. This implies differences in methods, scheduling, assessing progress, and evaluating individual contributors. Unless the organization is changed to be congruent with the imperatives of the technology, the likelihood of failure is dramatically increased.

Organizational deficiencies

Perhaps the most important area in which to look for the basis of project failure is in the area that has nothing to do with the technology itself: the organization. The critical consideration here is that technology adoption is a process that has a great many features independent of the technology itself.

The introduction of a novel technology changes almost everything that lies within the normal purview of management. Planning, work decomposition, scheduling, resourcing, and budgeting are all changed in ways that are difficult to deal with. Beyond this, the changes in work organization also change "political" relations within and between projects. The process requires a much more intentional and self-aware form of management.

Given these changes in management requirements, the opportunities to drop the ball are numberless. Enumerating them is interesting but ultimately uninformative. It is more useful to consider how to structure an OOT adoption project so as to maximize expected benefits.

IMPROVING THE ODDS

Know your problem

Perhaps the most critical step toward success in new technology adoption is a sound understanding of motivation. The rationale for it should be driven by an understanding of what the current problems are and how they might be addressed by the new technology. Adopting OOT merely because it is "hot" is not sufficient.There must be a connection between the costs or bottlenecks in current activities and the demonstrated advantages of OOT.

Once existing problems are understood, the next step is to show how these might be addressed by use of OOT and to state in terms that can be effectively measured what benefits are expected from it. Being able to say that you successfully adopted OOT is nice, but much better is being able to provide quantified evidence that adoption resulted in a cost reduction, higher quality code, or improved time to market.

Finally, OOT will be applied to a project. This project must be real. Unless OOT is evaluated in a setting that has tangible benefit it will never be seen as more than playing in the technology sandbox. Having said this, it is necessary to state that the project must not be a "make or break" project and that it must not be time critical. More on this later.

Know your technology

Essential to reducing the risk of failure is a full understanding of the technology being adopted. Several traps await the unwary in this area. One is to fall prey to external fads or internal zealots. For example, we all know that C++ is the most popular object platform. That does not make it the best for your problem. Another lies in viewing the technology selection as simply a matter of choosing the best language. This is too narrow. The technology must be evaluated in terms of its impact on every aspect of the project "from pipe dream to revenue stream."

One of the best ways to choose the most effective technology suite-analysis, design, programming, test tools, and so on, is to look at each technology in terms of every aspect of the product-development process. The relative strengths and weaknesses of competing technologies should be gauged for each aspect. Besides this, the importance of each aspect for the specific type of work foreseen should be assessed. Candidate technologies should be strong in critical areas and weaknesses can be discounted only if they relate to less important phases.

Know yourself

Organizational self awareness is very important in technology adoption. It is critical to understand how the existing organization functions and what its fundamental values are.

It is important that the organization be, in some sense, modern. Software engineering literature is very clear about the desirable methods and traits for effectiveness with established languages. A shop state of the art with existing technologies is far more likely to succeed with OOT. Indeed, self-evaluation may reveal that existing problems relate far less to the technology in use than to prevailing methods and procedures within the organization.

A critical part of being state of the art is the extent to which all elements of the organization make effective use of existing productivity tools. It does not matter whether these relate to design and analysis, project control, product testing, or software development. The point is that an already productive shop is far more likely to enjoy the productivity benefits of OOT.

Finally, it is important to understand the fundamental values of the organization and the kinds of behaviors that are rewarded. Is the organization bureaucratic in style? Does it reward grinds or cowboys? Above all, are those behaviors that are valued and rewarded congruent with the type of behavior most effective with OOT?

Small is beautiful

Assuming that you have done your homework and are ready to embark on OOT adoption, the most important consideration is the scale of the project to which OOT will first be applied. In general, smaller is better. In a word, run a pilot project.

The first advantage of a small project is that if failure does occur the cost is small. Equally important is that a team working on a small project is able to remain coherent and focused. Problems will arise, and the ability of each team member to remain constantly aware of the activities of others means trouble spots are less likely to go undetected and mutual support mechanisms can more effectively deal with problems when they do arise.

Team membership is critical. The players should be chosen for their flexibility and creativeness. Resilience in the face of problems is also important. Because these people will be used to seed further activity if the project is successful, they should also possess reasonable communications and technical leadership abilities.

Finally, the thing that distinguishes a pilot project from any other small project is that it is really a development project imbedded within a research project. The development project is creating a product and therefore lives within the normal project rules for planning, budgeting, making deliveries, and meeting schedules. The research project watches and evaluates the progress of the development project but does not participate in it. The one thing the research project can do that affects the development project is to freeze it when anomalies are observed. During this time, the development project clock stops and the budget belongs to the research project. Once the research activities are complete, the clock is restarted. This means that the development project may complete a six-month project in seven months and still be on time because the research project imposed a month of "stop-clock."

At the end of the pilot project, there are two possible outcomes: The development project either delivers or fails to deliver a product. The research project has a clear understanding of the effectiveness of the technology *within the organization*. On the basis of this, it is possible to decide whether OOT will be effective within the organization for solving further problems. The outcome may also be that either the technology or the organization is not yet ready for full-scale introduction of the technology. Whatever the outcome, an effective pilot project yields useful answers to critical questions and permits future avoidance of failure.

LOOKING FORWARD

One's initial use of OOT can never be risk-free, but a sensible and self-aware approach to the technology can limit the risks and bound the costs of failure. The pilot project approach is key to managing both risk and cost. Those factors that reduce the cost of a project in the first place also tend to reduce the chance of failure—and the cost of failure, should it occur. Speaking from experience, in the world of OOT it is the small successes that yield the big wins. ≡≡

Dr. Burton Leathers writes and speaks widely on the practical problems of OOT adoption. He is a Senior Consulting Software Engineer at Cognos, Inc., where he leads the Cognos Repository Development team. He can be reached at 613-738-1338 x3069, or by email at burtonl@cognos.

In the Industry

...Here's the bad news: the folks at Japan's Ministry of International Trade and Industry (MITI) are shifting their attention to software—long a Japanese weakness and a vigorous market for the United States. The good news is the MITI-funded software projects likely will include non-Japanese researchers. MITI has invited non-Japanese researchers to participate in projects before, but few joined because benefits from the nationalistic programs seemed limited. But these latest projects have more potential. After years of outside pressure, MITI is aware that openness is in Japan's long-term interest. It also hopes to benefit from outsiders' general software know-how. The projects target open systems and object-oriented programming that may help make up for Japan's deficiencies in software and are problems of interest to US companies. US and European researchers are expected to get some of the development money and can pitch their proposals to the administering committee (once it's established) of the Information-technology Promotion Agency (IPA). AT&T has already been in to present its ideas to MITI...

Japan's MITI shifts funding from hardware to software, Paula Doe, ***Electronic Business****, 1/13/92*

...Today, [Paul Morris, president of Paul Morris Personnel] emphasizes, employers are hiring and promoting only the currently needed skill sets. "If you know CASE tools and C++ and object-oriented methodology and you've implemented and delivered two large systems, that's the best insurance that an employer will take care of you..."

Recruiters: be selective, George Rostky, ***Electronic Engineering Times****, 1/20/92*

Languages

...A group of Computing Canada On-Line (CCO) users is planning to explore a C++ book starting Feb. 10. The idea is to follow a common schedule and discuss the concept raised in each week's "reading assignment." At press time, the text to be used had not been selected...

Learn OOPS on-line, ***Computing Canada****, 2/3/92*

...[Mark] Achler asserts that Whitewater's problem is less one of survival than of whether the company can distinguish itself from what Dataquest software industry analyst John DeArmon describes as "all the other boutique languages." With the continuing rise in Windows users, Mr. Achler points out that his company's prospects will continue to grow. "In a way, our future is linked to Microsoft," he says, pointing out that Actor was one of the first programming products designed to be used with Windows...

Software star thrashes in rough business waters, Arsenio Oloroso Jr., ***Crain's Chicago Business****, 1/13/92*

...Jane Siefel, project leader for a federal study of software production at Carnegie Mellon's Software Engineering Institute, estimated that some 70 million lines of Ada code are needed for near-term contracts. Given how long it takes to write and test Ada code, and the scarcity of competent programmers, Siefel estimated it could take decades to write the programs needed within five years. "For right now there is just an immense amount of software development to be done and there aren't qualified individuals to do it," she said. One plan, proposed by the Defense Advanced Research Projects Agency, which funds many high-tech research and development projects for the military, is looking into building a national library of Ada-based objects that defense contractor/programmers could use. With such a national repository of—presumably—tested and bug-free objects, programs could be cobbled together in a fraction of the time. In the meantime, large defense contractors have begun building their own in-house libraries of reusable objects...

A remedy for the software shortfall, Joshua Quittner, (Long Island, NY) ***Newsday****, 2/4/92*

...In traditional OOP environments, the end of a session, or the end of a job run, spells the end for the objects in a system. Memory is cleared and all objects simply disappear. Most OOP systems support interfaces to relational databases or PC database file formats. OOP should, however, focus all your system efforts completely, not just code development or running the program. To solve this problem, several companies now offer object oriented databases. I am not sure how well these databases actually meet the business needs of users. Many extend C++, a bandwagon that ran by me and I refused to board...Other databases work under the even more exotic environments of Smalltalk and LISP. With CLOS, the Common LISP Object System, LISP enters the object world with a standard. Objects existed for years in LISP, but no two companies could ever agree on how they should work or look. With CLOS we have a standard. MCC's Orion OODBMS project seized on CLOS and begot Itasca, both a software product and a company. Itasca is perhaps the best OODBMS on the market today, but who is going to write the next accounting system in LISP?...

The manufacturing mind: objectify your bandwagon, Daniel W. Rasmus, ***PCAI****, 12/91*

Case

...In the traditional four-step development cycle—analysis, design, coding, and testing—Rational's Rose focuses on the first two. Rose is one of the few, if not the first, independent tools for object-oriented programming that is not specifically linked to other products. Because Rose is independent, it can be used with C++ and Smalltalk, the two leading object-oriented languages, as well as Ada...

Rational grows a Rose without thorns, Howard Baldwin, ***UNIX World****, 2/91*

...Beware of CASE vendors bearing "object-oriented" gifts. Many CASE vendors are using OOP only to implement their graphical user interfaces. This does not mean they support object-oriented principles. In order for CASE technology to become truly object-oriented, CASE must adopt an entirely new approach to software design and development. Object-oriented principles of code reuse must replace current principles of code generation. Only then will the promises of object orientation, cost reduction and improved productivity be realized...

Beware of "object-oriented" vendors, John Jaakola & Daniel Pei, ***Computing Canada****, 1/20/92*

F.Y.I.

Excerpts from leading industry publications on aspects of object technology

Applications

...I'm impressed with the contributions OOP can make to commercial programming. But I'm much more impressed with how it's going to allow us to effectively personalize commercial, off-the-shelf packages...Just as programmers can build OOP-based applications by stringing modules of code like so many beads on a wire, you and I will be able to restructure programs by making menu choices that will also restring those beads...How important is this side of the OOP coin? So much so that in the end the impact of OOP will be measured less by its contribution to programmer productivity than to user productivity—gains that go more directly to how we use computers rather than how we build tools for them.

OOP will make OUR lives easier, too, Jim Seymour, ***PC Magazine,*** *1/14/92*

...Another spreadsheet in the Lotus stable is Improv... Developed in C++, it represents some pioneering concepts in object-oriented programming... Improv sprang from an idea someone had in 1986 for a better spreadsheet, he says. The company decided against restricting the design by making it compatible with 1-2-3. The difference between the two products centers around the modeling philosophy. "Data and formulae in 1-2-3 are tied to each other, particularly at the cell level," says Mr. Manzi. "In Improv the data is put in one place and the formulae in another, and you use fairly simple language to interact with the spreadsheet. Separating the logic of the formulae from the location of the data provides tremendous flexibility in modeling, which is unbelievably powerful. Some things we learned from Improv will find their way into other products. But there is not some verge purge that will occur in a few years where either Improv or 1-2-3 will disappear."...

Lotus to launch new Chronicle spreadsheet in '92, ***Canadian Datasystems,*** 12/91

...But confusion reigns about many applications of object programming. For instance, does attaching a CAD program to an object database make it object-oriented CAD? "No," says Scott Owens, marketing director of Spatial Technology Inc..."Object-oriented CAD systems are CAD systems that are written in an object-oriented programming language. Attaching a database to a CAD program can be done but that does not make it an object-oriented system."...

A computer without software accomplishes nothing, Jane Endrijonas, ***Managing Automation,*** *1/92*

ODBMS

...[Martin] Hardwick [president of STEP Tools Inc] says object-oriented databases have clearly demonstrated that they are capable of maintaining complex CAD/CAM files in an easily transportable format without a serious loss of system performance, at least on RISC workstations. "...The advantage of moving your data into an object-oriented database is that you get complete flexibility in accessing your data and manipulating it," Hardwick said. "And you can share it between all different CAD packages through IGES translation today and direct STEP tomorrow. It's in the interest of the customers, and the customers ultimately win."...

Open system philosophy takes hold in CAD industry, Aubrey McAuley, ***DESIGNNET,*** *1/91*

...David Maier, a scientist at the Oregon Graduate Institute of Science and Technology...has won a $2 million grant to design new computer software and applications for object-oriented databases. The three-year project will allow Maier and fellow researchers to develop a "query language" to make complex requests of the database for retrieving and updating information...

Lab notes [Vancouver, BC], ***The Columbian,*** *1/1/92*

Development Environments

...I'm one of those people who wears several hats. Being an end user of various software packages, I can appreciate the NeXT machine's capabilities as a host environment. Being a software engineer, I can certainly admire the many tools available for producing applications. But it is as a proponent of reusable software that I appreciate the NeXT machine the most, since I feel it is in this area that the NeXT machine's development environment leaves its nearest competitor in the dust....

NeXTstep and me, Andy Novobilsky, ***Object Magazine,*** *1-2/92*

...It became apparent to me that the Defense Department is interested in these sleek-looking black [Next] computers when I saw lots of uniformed servicemen at the Next federal expo, mostly Air Force. But I had to ask myself, what does the DOD want with a workstation that has very little security—at least in terms of the operating system—and is not Posix-compliant? I thought those were the two big requirements for DOD workstations. Federal Next system users must not think so...So if it's not security, what is it that these federal Next users see in the machine? Application development. The Next machine, with its NeXTstep object-oriented system software, is said to be a software developer's dream come true....

Who knows what's Next in the Fed arena, Darryl K. Taft, ***Government Computer News,*** *1/20/92*

Engineering Methods

Mitigating madness with method: first establish what you value

by Michael Fuller

The problems plaguing the software development community have been extensively researched, reported, and consulted on for over 20 years.

While there is no general agreement on how to solve these problems, there is an evolving consensus on the desired objectives. I think we who apply information technology can agree on these objectives:

- develop software components that can be reused by many systems
- leverage the experience of the most productive developers
- document the system requirements in a way that allows end users to verify the results will meet their needs
- identify and correct problems early in the development cycle

The critical element to achieving these objectives is to articulate the values that will govern the daily life of our users, managers, developers, methods, and systems. History provides numerous examples of value systems that have withstood the test of time and teaches us that shared value systems are the foundation of cultures and social groups. A manager responsible for selecting and implementing a new technology must first understand (or establish) the shared values of the group into which the technology is being imported. These values are a basis against which to evaluate technology alternatives and appropriateness. While the specific values may vary from organization to organization, effective value systems must:

- define first principles
- define standards that determine when a principle has been violated
- establish an ultimate authority with the power to make and enforce a final decision
- include a way to modify decisions, standards, or principles to adapt to an evolving world

The former computer manufacturer Fortune Systems provides an excellent example of the problems caused by conflicting value systems. Fortune was chartered to develop an inexpensive information appliance that could be used in every business office. Management's value system emphasized high margins, low manufacturing costs, and quick time to market. Marketing adopted values that made business applications, no end-user training, and a low selling price paramount. The hardware developers decided to meet these requirements by designing proprietary hardware based on their values of highest possible performance at the lowest possible price. The software developers responded by choosing UNIX as their operating system, cloning the Wang word processor, and contracting for a proprietary menu and forms-driven user interface to hide the complexity of the entire system. The subsequent failure of the product in the market place can be attributed to management's failure to ensure that the value systems of marketing, hardware engineering, and software development were in sync with its own values—and that all concerned were actually developing the same product! Management targeted the floppy-based IBM PC marketplace with a price point of $3,000 while the actual customers were ordering systems with 20 MB hard disks at a price point of $12,000. Clearly, internal definitions of "inexpensive" were contradictory. Furthermore, marketing wanted a turnkey system requiring limited support, so it targeted large corporations, but large corporations were not in-

Michael Fuller is an information engineering consultant and has developed large-scale, distributed applications with a variety of technologies including Eiffel. He can be reached at 415.928.7067.

terested in a start-up company using an unknown operating system because they made low risk a priority in their value systems.

The failure of Fortune Systems to adopt a consistent set of internal values up front that was in sync with their intended market resulted in confusion and, ultimately, bankruptcy.

WHY SHARED VALUES?

A value system provides guidance and acts as a benchmark. It provides the foundation required to make objective decisions in a world often short on objective facts. To operate successfully with information technology, we need a clear understanding of the values we hold and the values we can provide to our clients and users. What we need is a bridge between the abstract world of our values and the concrete world of the systems we develop. We need this for success in adopting new technologies like object orientation, for implementing software development methods, and just for the "simple" task of developing useful, usable software applications.

Let's focus on software development methods. We are talking about methods for disciplined teamwork. How do we get a group of skilled professionals to solve a problem too large for any one person to fully grasp? We need a way of identifying, planning, and judging what work steps and intermediate products are needed to ensure that the end product of our effort is consistent with our value system. We need this whether or not we consider our developers "object oriented." Thus, our chosen software development method must support and demonstrate our chosen values, and should include:

- goals that describe the expected results
- strategies that describe how each goal will be achieved
- critical assumptions that define the context or basis for the chosen strategy
- critical decisions that identify the commit points

The individual components of the method provide a context that permits matching of our value system with business reality. If the candidate method supports our value system and is appropriate for our environment, then it will provide an excellent start toward developing superior-quality software.

How does a method deliver on such a promise? By organizing and managing some of the complexities of software development, a method allows the developers to concentrate solving the business problem. Are these results guaranteed? Only by those individuals not responsible for the results. What can a method accomplish? It can provide the implementation plan for your value system. Coupled with common sense and a sense of humor, an appropriate method will:

- provide a framework for project planning, scheduling, and resourcing
- provide context for each task
- measure and evaluate progress
- provide a common language for identifying requirements, defining terms, and making decisions
- provide basis for objective criticism of work results
- provide a way to leverage the skills of the development team

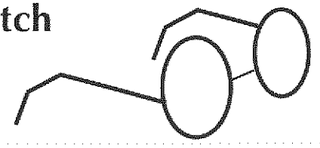
A method is a tool for managing a process, not a solution to a problem. I cringe every time a project meeting is consumed by argument over whether we should start by defining the data, the process, or the events. A method should reduce these discussions to their core: "How will we manage the risks inherent in this project?" The expected work products and the criteria for judging them are part of the definition of the method. The specific techniques used to achieve these results should be driven by the preferences of the team members and the nature of the problem. Techniques are not specific to any particular method or technology and provided that the requirements and expectations of the method are satisfied their use should be a matter of personal choice.

To be most effective, the set of techniques used should be interlocking. The results of from the current step should build on and verify the results of a previous one, and should support the later steps. Adrian Bowles stated in a recent OBJECT MAGAZINE article[1] that using an interlocking set of techniques is an invitation to "analysis paralysis." He argues that it is better to concentrate on defining and implementing the elements of the universe (objects) and then connect them to implement the required system. If attempting to understand the nature of an organization, the challenges it faces, and the information and activities in place or needed leads to failure, then how will not confronting these issues lead to success? Is there a single example in any area of human endeavor that relies on not planning and a lack of understanding as the most effective technique for solving a problem or creating a product?

Bertrand Meyers argues that the programmers and their managers are being forced to deal with problems that are beyond the capabilities of their current techniques. He believes that software quality should not be compromised, and that the way to prevent this is to build software whose qualities can be defined and measured. Meyers proposes a set of values for developing quality software as follows:

- *Correctness*—the ability of software products to exactly perform their tasks, as defined by the requirements and specifications.
- *Robustness*—the ability of software systems to function even in abnormal conditions.

Book Watch



The "Book Watch" column does not contain book reviews. These listings are abstacted from press releases provided by the publishers, and no endorsement is implied. Please send announcements to the Editor: Robert Shelton, 1850 Union Street, Suite 1584, San Fransisco, CA 94123; Fax: (415) 928-3036.

Developing Object-Oriented Software for the Macintosh

Neal Goldstein and Jeff Alger
Addison-Wesley Publishing Company
Reading, MA; 617.944.3700, x2278.

The first book to deal with the complete process of developing object-oriented software, from analysis through design and programming.

Designing Quality Databases with IDEF1X Information Models

Thomas A. Bruce
Dorset House Publishing
353 W. 12th St., New York, NY 10014; 212.620.4053

This new publication describes how to use information models to specify business requirements, policies, and rules, and details how to use these specifications to design and build high-quality database applications. This text represents the first definitive work on the theory and practice of data design.

The Decline and Fall of the American Programmer

Prentice Hall Simon & Schuster Education Group
Professional Technical & Reference Publishing
113 Sylvan Avenue, Route 9W, Englewood Cliffs, NJ 07632
201.592.2348

In his new book, Edward Yourdon demonstrates how US software organizations can become world-class shops if they exploit the key software technologies of the 1990s. The most important chapters in the book deal with "peopleware" or management issues crucial to running an efficient software development operation. Technologies that are the key to survival are reviewed.

... continued from page 16

mation they need and documenters must provide alternative routes to that information.

The index is possibly the most important part of a reference manual. And every manual, whatever else it may be, will also at times be used as a reference. The table of contents reveals how the main text is organised. It is important that the index should provide alternative access routes when the table of contents is unhelpful. So if the table of contents has verbs as chapter headings, the index should be particularly rich in nouns—not only the names of things in official "systemspeak," but the names they might be given in everyday language. If the table of contents is organised by nouns, the index should pay special attention to verbs denoting the user's possible or desired actions.

Users of computer software need to understand the actions and objects that will help them accomplish their professional tasks. An effective user has a good mental model of the system, and objects are the natural components of such a model. User manuals tend to concentrate on actions, and should pay more attention to the objects of the system. This will become particularly important as object technology is placed in the hands of end users. The principles of task orientation and minimal design remain valuable to the documenter, but may leave the user without an adequate answer to the question: What have we got here, and how does it work? ≡ ≡

Acknowledgments

The author thanks Elizabeth Churchill (University of Nottingham), and Thomas Green (MRC Applied Psychology Unit, Cambridge) for their assistance with this article. However, they are not responsible for its content.

References

1. Halasz, F.G. and T.P. Moran. Mental models and problem solving in using a calculator, PROCEEDINGS OF ACM SIGCHI, 1984.
2. HP NEWWAVE USER'S GUIDE, 3RD ED., Hewlett-Packard, 1990.
3. Brockmann, R.J. WRITING BETTER COMPUTER USER DOCUMENTATION, FROM PAPER TO HYPERTEXT, VERSION 2.0, Wiley, New York, 1990.
4. Norman, D. THE PSYCHOLOGY OF EVERYDAY THINGS, Basic Books, 1988.
5. Micronta LCD 150' Water-Resistant Watch, Catalog No. 63-5061, Tandy Corporation, 1988.
6. LOTUS AGENDA USER'S GUIDE, Lotus Development Corporation, 1988.

product complexity increases, the end products no longer have the clean modularity that we associate with good object-oriented systems. Of course, today we aren't dealing with hardware at all but with software that embodies every possible error in user-interface design. Multiple functions are heaped on each control button, in the manner deplored by Donald Norman.[4] The effect of each control depends on the mode in which the machine is currently operating. Often there is no visible indication of the mode nor any reliable means of escape back to a known mode. Modes or states—rather than parts or objects—become the prime opportunities for hierarchical decomposition.

The instructions for digital watches—another excellent example of user-interface complexity—are meticulously task oriented. A typical example has seven "recipes" of up to seven steps each.[5] Learning these recipes by rote seems out of the question. Significant intellectual effort is required to form a mental model of the device. It may be helpful to sketch the chronometer's state-transition diagram, using a pencil or your favourite CASE tool. An STD provides a behavioural model of the watch. Structural models can also be useful. A calculator may be easier to use if one has a mental picture of its internal registers.

MENTAL MODELS

The existence of a mental model implies that we have an understanding of the system that goes beyond procedural, "how to do it" knowledge. The mental model is "runnable" and can be used to explain and predict the system's behaviour. A mental model may also be useful in devising a sequence of actions to carry out a novel task.

It is unlikely that anyone relies on procedural knowledge alone. Probably everyone forms some kind of mental model of a system. People who seem to have no mental model may really have inaccurate or incomplete models. If so, training and documentation should attempt to encourage novices to construct a useful and broadly correct model of the system and help experienced users to extend or correct their mental models.

CONTROLLING COMPLEXITY

Object-oriented models are useful to the software developer as a means of controlling complexity and reusing ideas. The same mental models should also be valuable to the user of the software, for the same reasons. Documentation can give the user access to the mental models that helped the software designer.

Users probably won't remember complicated inheritance hierarchies, but the software will be easier to document (and easier to use) if related classes of objects have similar behaviour. As the designers of the Apple Macintosh understood, the biggest win is reuse of the user's knowledge. This is achieved by ensuring that objects of different types behave consistently within and between applications. Consistency in the user interface—and in the underlying conceptual model—means there is less to learn, less to go wrong, and less to document.

DOCUMENTING AGENDA'S OBJECTS

Some authors have usefully adopted an object-oriented approach to documentation even when the software itself is not object oriented. Lotus Agenda is an application that can be used in many different ways. In that respect, it resembles the object software we'll use in future years. Some people will get the hang of it by working hands-on, with concrete examples. For them, Lotus provides a six-lesson tutorial in about 100 pages.

An entire manual based on the "creating a document" approach would probably not have worked, because Agenda users all want to do different things. Instead, the reference manual[6] begins with four task-oriented introductory chapters, and then goes on to deal with Agenda's entities one by one. Here are some exemplary chapter headings:

- Working with Items and Categories
- Working with Sections
- Working with Columns
- Working with Views
- Working with Categories
- Working with Dates
- Working with Notes

People who find Agenda hard to understand should switch off the computer, settle down with the manual, and sketch an entity diagram for the application as they read. Honest, it fits on the back of an envelope! Underneath Agenda's rather complex command structure is a simple but powerful conceptual model. Once you grasp the conceptual model you can begin to think about the ways you could use Agenda. For abstract thinkers this is probably the best way into the package.

NOUNS, VERBS, INDEX, AND TABLE OF CONTENTS

The object-oriented approach to documentation is not necessarily suitable for all software or all parts of a program's documentation. Documentation has to serve the needs of users who vary greatly in their professional experience, computer experience, and learning style. Readers will use many different tactics for finding the infor-

continued on page17...

- *Extendibility*—the ease with which software products may be adapted to changes of specification.
- *Reusability*—the ability of software products to be reused in whole or in part for new applications.
- *Compatibility*—the ease with which software products may be combined with others.

This is a set of values for software that I feel is—or should be—universal. It is equally applicable to business applications and systems programming, batch and on-line transaction processing, centralized databases and distributed client–server architectures, procedural and object-oriented technology.

TOTAL COST AND TOTAL QUALITY

It would be convenient if success in the information technology business was achieved simply by a simple recipe: Articulate your value system, select a method, apply the techniques, stay vigilant, and ride a white horse into the setting sun. While success will elude us until such basics (excepting, perhaps, the white horse...) are accomplished, even covering the basics does not guarantee success.

A fundamental challenge remains: to identify and justify the costs associated with adopting our chosen values, methods, techniques, and support systems. Application of information technology is, after all, a business decision.

The most obvious costs are financial; the more significant ones are not directly monetary. Even in the category of monetary costs, some surprises await—hidden costs like:

- *Software tools*—beware starter kit vs. regular pricing on the workbench tools. One vendor charges two to three times more for the "regular" copies.
- *Workstation hardware*—comprehensive tools work best on high-end systems.
- *Central repository or encyclopedia*—this "optional software" normally contains the functions needed to reconcile the work of multiple developers and manage multiple versions.
- *Classroom training*—attending formal courses is a quick and relatively inexpensive way to shorten the learning curve. If given a chance, I would attend the training *before* buying the tool.
- *On-the-job training*—the method and techniques are most effective in the hands of experienced practitioners. The first few projects may take longer to produce quality results than promised by the vendor or expected by the developers.
- *Assumption of new host hardware and software*—many consulting practices, magazines, and seminars are pointing to the existing data centers as the primary cause of problems, recommending as a solution to "throw the bums out" and buy from one of our open-systems consortium technology partners. Beware the conflict of interest that outweighs your best interests.

Complicating the financial decision are the problems of justifying the expense of implementing a method and its supporting infrastructure using traditional financial measurements. The fundamental problem is that most organizations cannot detail the current cost of producing software that precludes any statements of how much better things will be by any measure: dollars saved, return on investment, reduced time to market, or improved programmer productivity.

> ***A method is a tool for managing a process, not a solution to a problem.***

The approach that seems reasonable and repeatable is rooted in continuous process improvement with customer satisfaction as the benefit expected. While identifying and measuring satisfaction is just as problematic as measuring costs, the methods and techniques being advocated for software development borrow from the total quality community. This raises an opportunity for fundamental improvements in the processes of software development to ride the justification of improvements overall in the business.

Besides the direct and indirect financial costs you will incur, there is the problem of time. The problems you are trying to solve took years to develop and everything surrounding them will resist change. It will take time to understand what the problems are, identify the root causes, and build a consensus for solution. The technicians, users, and managers will need time to develop the necessary personal and technical skills. It will take time to realize the benefits, but the expense will be immediate and continuing.

THEN THERE IS RISK

The final obstacle is managing the risk inherent in changing any entrenched system. The most personal consequence of identifying the cause and proposing a solution to fundamental organization-process problems is akin to painting a not-so-imaginary target on your body. You will be challenged to justify your ideas by demonstrating that the current practice won't continue to fail. A brief sample of my favorite red herrings in this area are:

- This stuff is just another technical fad.
- The techniques and tools won't work as well as you expect.
- The grand alliance between companies X, Y, and Z is certain to fail.
- I read in yesterday's newspaper that this is old hat, proven not to work.
- This technique is unproven; just look at OS/2 (or UNIX, DOS, Windows, etc.).
- We will train all these people and they'll quit to take jobs elsewhere.

All these complaints translate into "We will not commit to the success of this endeavor." Unless you can earn and keep the support and active participation of all levels and groups within the company, the outcome will be driven more by emotions than common sense. Machiavelli's classic warning has been proven beyond a reasonable doubt: "It must be remembered that there is nothing more difficult to plan, more doubtful of success, nor more dangerous to manage, than the creation of a new system. For the initiator has the enmity of all who would profit by the preservation of the old institutions and merely lukewarm defenders in those who would gain by the new ones."

> ***It will take time to understand what the problems are, identify the root causes, and build a consensus for solution***

I have watched several companies attempt to implement methods, techniques, or tools and have yet to see a success that was driven by traditional programming staff or their management. The risks were too great to gain the level of support from the other communities needed to ensure success. I have seen two cases where the users were able to achieve a foster success.

The first case began with two people starting a backroom project to document the current data files. Over the course of three years, this group gained a reputation for understanding the business systems and became the honest broker between the various development groups. This was never an official position but those teams who accepted their advice produced systems that were less troublesome. Ultimately, it was simple peer pressure that brought the development staffs and their many systems into compliance.

The second case was more direct. The president of Xerox Corp. visited Japan in the mid-1980s seeking evidence to support charges of anti-competitive dumping practices by his Japanese competitors. Surprisingly, he got exactly what he asked for. As the saying goes, it was under his nose all the time. What great truth was discovered? The old one about "We have met the enemy and they is us." He found strong evidence that the Japanese competition simply did a superior job, built a better product, and did so at a lesser cost. In short, they better understand their values and implemented processes to achieve the desired results. For 99 out of 100 companies the results would have been predictable: "A corporate-wide initiative to reduce costs and improve productivity...." In this case, the company president did something out of the ordinary. He told his staff through words, action, and incentives that things had changed—and results were expected. For MIS, this took the form of having to meet the same productivity, quality, and cost measurements as the other corporate operating divisions. This is an example of riding the coattails of a serious corporate quality effort.

THE BOTTOM LINE

As a manager you are expected to evaluate, select, and implement new or improved technology—successfully. As a subscriber to the Hotline, you have at least implied a need to understand and track the evolution of object-oriented technology. Whether you are a strong supporter, harsh critic, or skeptic, the basic issues are the same: Is this something we should be using? If so, how? Where? How can it be implemented successfully?

How can this question be answered in the absence of an established set of values against which to judge the technology and your implementation alternatives? Without first deciding on common values, a new technology cannot be successfully absorbed and implemented. Without established values, it is unlikely that a mechanism (read method) for successful and repeatable use of the technology can be instilled in the organization.

Your users, staff, and customers must first define the software equivalent of the Bill of Rights or the Hippocratic Oath. Every proposal, project, and situation can then be measured against these principles and judged on their merits. Any changes or adjustments should be obvious, or at least acceptable, to all concerned.

I have never understood why software has for so long been considered so unique that the laws of God and man don't apply to its development. Until it is treated as every other complex human endeavor, its creators will continue to fail like nothing before or since. ≡≡

Reference

1. Bowles, A. Objectified information engineering–the method time bomb? *Object Magazine* 1(6):23, 1992.

to educate the user about these objects, rather than merely to inculcate a set of procedures.

TASK-ORIENTATION: NOT THE WHOLE STORY?

Some documentation authors may feel that in writing explicitly about objects, they are paying too much attention to the internals of the software while ignoring the job the user has to do. Certainly documentation authors should not forget what they have learned about task-oriented documentation, but a document or other data object that the end user has created while using the program hardly counts as a software internal.

If documentation readers are to become effective users of a computer system, they need to form a mental model of that system. Researchers from Halasz and Moran[1] onwards have studied how people learn to use simple devices such as calculators. It turns out that people perform better if they are not merely told how to use the device but also how it works. This is the crucial component that is often missing from task-oriented manuals and suggests a need for object-oriented documentation.

The NewWave manual[2] is a good example of task-oriented documentation. Instead of describing the software's features one by one, it introduces features as they are required in the course of realistic tasks. Sections have titles like "Working with objects" or "Creating documents." Each section contains a short introduction followed by maybe a dozen recipes for carrying out simple tasks such as copying an object or creating a text note. Most recipes are less than two pages long, including screen shots and a step-by-step summary. NewWave also offers extensive online help.

The case for task-oriented documentation is eloquently stated by John Brockmann,[3] who argues that users dislike feature-oriented documentation that "forces readers to center their business duties around the software rather than vice versa"—an argument that sounds strikingly like the case for object technology (OT) itself.

Brockmann notes that tutorial and reference manuals receive different treatment. Tutorials take a step-by-step approach and are usually organised around user tasks or a hierarchy of user needs. Reference manuals are organised like dictionaries or encyclopedias. Brockmann argues that both kinds of manual should be task oriented. The text of a reference manual should say what the software can do for the user. The index is the place where commands and features can be listed systematically. Some experience supports this. End users surveyed by Microsoft said they wanted a command/feature-oriented index as well as a task-oriented index.

Brockmann believes that reference manuals should assist the user who wants to combine product features to do something original. This implies that task-oriented manuals can be more than mere recipe books. He is aware of the manipulative, deskilling connotations of task-oriented documentation. It was, after all, the notorious Frederick Winslow Taylor who used "task work" and instruction cards to attack the skilled craftsmen in the steel industry a century ago.

MINIMAL MANUALS

For the software documenter, the suggested antidote is to follow the minimalist design philosophy proposed by John Carroll of IBM. A minimal manual tells readers only what they need to know to get started on productive work. It encourages exploration by deliberately giving incomplete information. The reasoning behind this is that people are impatient to do some real work, and won't read a manual cover to cover, preferring to experiment and learn from their mistakes. This, of course, presumes that the software allows the user to gracefully and harmlessly recover from mistakes.

Both task orientation and minimal design are based on a belief that people learn by doing. But are tasks and actions the only things to be documented? What about objects? Only at the level of machine code is software literally a list of actions. People will encounter user-interface objects such as windows and icons and data objects such as documents. Should not the manual deal with software's nouns, as well as its verbs?

Object-oriented systems are likely to require object-oriented documentation. Tasks won't disappear from the scene. But every task involves both actions and objects. In traditional procedural software, the essence of a task could be captured by breaking it down into actions. Future software will work with much richer and more varied data structures, with a lot of complexity that simply isn't captured by task-action mappings.

Even if they don't abandon task orientation, documenters may need to pay more attention to objects and proportionately less to actions and procedures. Learning by doing has its limitations. It is hard to build a mental model of any complex thing purely by observing its behaviour. If readers are given a direct description of the objects involved in software, perhaps they will be able to build a mental model more rapidly.

LESSONS FROM HARDWARE

Object-oriented software isn't like other software. It is more like a physical device. When you are trying to understand a stereo or a lawnmower, one of the most useful things in the manual is an explanation of the parts and their functions. At least, that used to be the case.

Unfortunately, physical devices like video recorders and fax machines are becoming incomprehensible. As

SOFTWARE DOCUMENTATION

Objects and actions in end-user documentation

by Tony Durham

The object paradigm is transforming the way software is developed, and in due course it will transform the way software presents itself to the end user.

The manual writer is only concerned with software as it appears to the user. It matters little whether the software was written in C++ or in Basic. As yet, the job of the documenter has barely been affected by the advent of object technology. That may be about to change.

Object technology is still perceived primarily as a developer's technology, with benefits in software quality, productivity, and maintainability. But end users are beginning to experience the power of the object paradigm for themselves. The experience of manipulating objects, and constructing complex objects from simpler ones, is available both within applications (such as MacDraw) and at the system level (as in NewWave by Hewlett-Packard [HP]).

Admittedly, office workers are not usually entrusted with the full power of object-oriented programming. For the most part, it remains the programmer's prerogative to define new classes and use inheritance mechanisms. Much end-user programming is procedural, and consists of writing scripts to automate repetitive tasks. Examples are Lotus 1-2-3 macros at the application level and NewWave agent tasks at the system level.

Apple's HyperCard represents a brave attempt at letting nonprogrammers program with objects, messages, and a form of inheritance. Ironically, the reader of a HyperCard stack gets little sense of manipulating objects; instead, it's the old, old story of navigating from screen to screen. HyperCard probably gave many Macintosh users their first experience of getting badly lost in computer software.

Tony Durham is a freelance journalist and information technology consultant based in London, England. He monitors object technology as the European Correspondent for the HOTLINE *and our sister publication* OBJECT MAGAZINE. *He can be reached at +44 81 960-0536.*

BRAVE NEW WORLD

Despite its uncertain progress to date, the objects-for-everyone idea is widely assumed to represent the future of application software. Separate, monolithic application programs are supposed to disappear in the brave new world frequently alluded to by Microsoft's Bill Gates and Borland's Philippe Kahn. Instead of issuing commands to an application, the user will manipulate objects of different types (text, graphics, images, worksheets, audio, video) that behave in familiar and predictable ways under all circumstances. Thus, you won't use a text editor: Text, wherever you find it, will be intrinsically editable. [*Ed: hmmm… newsletters that edit themselves…*]

HP's NewWave provides a foretaste of this within a DOS/Windows 3 environment. NewWave is still a powerful piece of software, and at it's launch in 1987 it seemed futuristic. It encapsulates files, directories, applications, and utilities so that the user instead sees easily managed objects, folders, and tools. Bits of the old world still coexist uneasily with bits of the new world. Start up a DOS application from within NewWave, for example, and you are dumped back in the old world until you exit the DOS program. But enough has changed to prompt the question: What kind of manual is most helpful to the user of an object-oriented system?

HP Laboratories in Bristol, England has done extensive research on object-oriented office environments. Its work goes well beyond what we see in NewWave. But an inquiry about research on user documentation drew a blank. HP Labs develops the technology. The business divisions document it and sell it. HP does not seem to have made the connection between innovation in software and innovation in documentation. For that matter, few other manufacturers have either.

Here's a possible scenario that makes the connection. End-user software will become increasingly toolkit-like. The user will need an understanding of the objects and classes in the system, plus the imagination to invent new ways of using them. Documentation will need

PROFESSIONAL DEVELOPMENT

Championing object technology for career success in the 1990s

by K.L. Straat

Object-oriented technology could be a bobsled to success for your company, your work group, and for you personally.

The time is right to champion OOT in your company. What a tool for recruiting others around you, getting involved in new projects, meeting new people, and gaining a reputation as a "doer!"

But what about this bobsled? Why OOT, and when and why success?

For years, the standard career dream has been one of a steady ascent to a high pinnacle, the nature of which was chosen by each person individually. Reality made that path more like a roller coaster with sharp and unpredictable ups and downs—and the 1980s made it like a bobsled on an icy track! Ready for the 1990s?

Acknowledging that the steady ascent is a dream, let's look at careers a different way. Compare the roller coaster with the bobsled. A roller coaster is run by a professional. You take a seat in the car and are strapped in. The ride begins slowly and continues with a breathtaking drop and a series of jolting curves and dips that ends by coasting to a stop. A roller coaster career is dynamic but reasonably predictable, and the employee leaves with a gold watch and retirement benefits.

The bobsled better describes careers today. You choose to play because you want to. You buy your own sled, find your own team, and train together. Each team member depends on the others for success. The track is icy and variable in its long course, filled with curves and ruts from other sleds, with walls that are as much an invitation to danger as they are a protection. You push off with all your strength, jump on the sled, get your best aerodynamic position, and carefully steer into the first curve. You're in control, responsible for the outcome. The sled will not make it without your guidance. You have to recognize this to succeed in your career today. This is career direction in the 1990s! You're in control. Predictability is what you make it. Few seat belts exist, and certainly there is no fixed track to assure a safe finish.

In the work environment, nothing is like it used to be: not careers, companies, management, or jobs. No doubt you know that you have to take control of your own career direction. But if you're involved with OOT, the fast track is coming to you.

OOT is poised for an S-curve of growth. The sharpest part of that growth is right in front of us—a great career-enhancing opportunity. Here are some indicators that support my reasoning:

- OOT is now commercial. It was conceived in the mid 1960s, nurtured in the 1970s, optimized in the 1980s, and is entering the 1990s with a substantial infrastructure.
- OOT has major successes to its credit. A few examples are: big telephone switches by AT&T, CAT-scan examinations by General Electric, emergency-room equipment by Hewlett-Packard, and MIS at Brooklyn Union Gas.
- CEOs are talking about OOT, and more CIOs are saying that they are exploring OOT with plans to do more in the future.
- An industry association, the Object Management Group, is setting standards so that competing companies don't confuse the marketplace in their eagerness to close sales.
- Class libraries are being recognized as a new resource special to OOT, and more importantly, these libraries are still reasonably priced, if not cheap, for the value added.
- The ease of maintenance of OOT programs is being proven by experience.
- Professional vendors of software—AT&T, IBM, General Electric, Texas Instruments, Hewlett-Packard,

Kent L. Straat is a partner of Object Resources International, a search firm dedicated to all disciplines in object technology. Recently Vice President of Staffing at Gartner Group, he has been in the search field for 19 years following experience on Wall Street and with IBM. He is the author of What Your Boss Can't Tell You—*how to evaluate your company, your job, your goals, and your performance. He can be reached at 203-325-1919.*

Allen-Bradley, Microsoft, and Apple—took up OOT before the MIS departments. Early commercial users were equally impressive: Boeing, Siemens, GTE, Northern Telecom, Dun & Bradstreet, Schlumberger, and McDonnell Douglas, for starters.

- OOT concepts are being incorporated in open systems standards like the X Windows system, and in emerging technologies like hypermedia and multimedia.
- For C++ and Smalltalk together, more than 750,000 licenses have been sold. That's impressive, even allowing for shelfware.
- Gartner Group predicts that as much as 90% of PC and workstation applications will be developed with OOT by the year 2000.
- Journals, magazines, and directories are dedicated to OOT as are annual conferences drawing up to 2,000 attendees. An OOT meeting directory lists some 50 events worldwide, among which are OOPSLA, C++ World, LOOK '92 in Denmark, OOP '92 in Munich, and OBJECT EXPO in New York and Europe. Over 30 books on OOT are now in print.

...you'll definitely want to avoid those who have jumped on the buzzword bandwagon so they can tap a new market for what they have always sold.

That's one persuasive list on the positive side of the OOT balance sheet, but the negative side has some entries that must be considered. Just acquiring the mindset required to succeed with the technology, and getting people trained, takes significant time. The first application a team builds using OOT often takes more time and costs more money than an equivalent conventional-technology product—even if you do have class libraries and effective development tools to leverage your trained team. Experience shows that getting skills up to an acceptable level can take between 3 months and 2 years, depending on the skills of those involved, the training provided, and the tools available. And mastering the concepts of a graphical user interface (GUI)—almost a prerequisite, and a given for object applications—can be a significant challenge for the newly initiated.

But the balance sheet favors OOT as a key technology likely to burst forth in the 1990s. The question, then, is whether you have the confidence in yourself and in the direction of this technology to stake your career direction on this industry trend. Remember that you will have to drive, because your career is your bobsled.

So, why is there success for you in championing OOT in your company? To put that in perspective, let's again look at the indicators as we did with OOT. Career success requires a broader outlook than simply achieving success with a technology. Tom Peters of *In Search of Excellence* and *Thriving on Chaos* fame said, "Predictability is a thing of the past" and also: "We are in an era of unprecedented uncertainty...to win tomorrow requires quality and flexibility." Richard Pascale, author of *Managing on the Edge*, said that stability is a dangerous mindset for a company. It stifles creativity and risk taking. We have entered an era where taking risks and becoming a leader is crucial to survival as much as success. Witness Ted Turner, *Time* Magazine's 1991 Man of the Year, founder of CNN, resuscitator of the Atlanta Braves and America's Cup Skipper. He has a sign on his desk saying, "Lead, follow or get out of the way." Such a subtle gentleman.

Rosabeth Moss Kanter wrote in her book, *When Giants Learn to Dance*, "You are known by the quality of the projects of which you are a key part, and not by your title in a hierarchy." Picking up Kanter's lead, let's think about what this means for projects using object technology. Because the industry trend suggests that OOT projects in the 1990s will increasingly be high impact and highly successful, the obvious alternative is to associate yourself with such projects.

In 1988 I wrote *What Your Boss Can't Tell You* to help professionals through a review of the myriad factors that can affect a career. A periodic review is required to see if anything might be about to throw your career off track, or to see if key opportunities are available to be tapped. This is where OOT comes in. OOT is an opportunity, and sticking with non-OOT technology alone can set you back in your career. OOT simply provides an opportunity to be a part of successful, high-visibility projects.

Championing OOT requires that you know it well: history, details of implementation, benefits, how to proceed, and above all, how to keep it simple. Remember that nothing fails like enthusiastic complexity! In describing OOT to others, when selling it to your peers, managers, and team, remember that OOT is a new concept to most professionals, and technological complexity can be perceived as a threat if not made understandable.

As for history, you should know who has undertaken projects in your field and what the results were. Get to know the movers and shakers in the OOT field. Attend conferences to meet others using and managing the technology and commit enough time to discern the experts from the hangers-on. You may want to call upon the experts—or they may want to use you as a resource—and you'll definitely want to avoid those who have jumped on the buzzword bandwagon so they can tap a new market for what they have always sold. Get in the loop! Communicate and learn from the hard-won experience of those who have gone before!

For implementation, know the steps of the process and the resources upon which you will draw. How are you going to determine the requirements of the project? How do you coordinate the analysis, define the objects, classes, hierarchies, and determine the right methods to use? Emphasis is often placed on prototyping and the iterative reviews with the users of the product, but knowing how to manage all these factors is crucial to success. We stress human factors in designing GUIs, but getting everybody in a development organization proficient with an agreed-upon set of standards is paramount—and a challenge.

As to the benefits of OOT, know to whom they are the hot buttons, and sell to these individuals. Lay your groundwork. Make sure you have strong support behind you at the outset and throughout each project you undertake. Quantify the benefits in projected savings or added revenue or shorter time to market. The savings could be in dollars, time, or maintenance as projected new releases are deployed. Time saved in providing customer service or introducing new products is not only money, it is competitive advantage in the marketplace—positive achievement for your company and your own career as a result.

Selling OOT and organizing your team to deliver on the promise will put great demands on your human relations skills. Know who your management sponsor is—the person who has the authority to approve, fund, and support your effort, and has an interest in doing so successfully. Your sponsor must initiate the project and understand the ramifications of doing so—and must stand behind you as difficulties develop that must be overcome. You yourself have to know the ramifications to educate your sponsor, and be cognizant of what you are asking for when getting project sign-off. You also have to know those people who will have to change as a result of your project, and those groups that will advocate your project but don't have the clout to be sponsors of it. Being in control of management ramifications is basic to project and career success. If this sounds like becoming political, in the constructive sense of the word, you're right. Part of career success with OOT is success with people. OOT is not a technology in a vacuum.

Introducing change forces you to anticipate the impact of that change and the likely resistance to it. OOT represents change. How extensive are the effects of the change? Are people motivated to change? Will they have to change their values? Will job procedures or equipment utilization change? Whose personal interests will be challenged? What social relationships will be changed? How might job evaluations and performance incentives be changed? Thinking in this framework will help you smooth the transition to OOT because it will strengthen your ability to help yourself and others manage the change.

This kind of technological change is the subject of a new body of study called sociotechnical systems, or STS in the jargon of the organizational development people. STS considers the effect on social patterns in the workplace caused by changes in information systems (IS) technology. Championing OOT means driving technical change not only in the social workplace of the company but also in your own IS group. It is imperative to be sensitive to the social changes caused by the introduction of new technology. Even if *you* think the changes are minimal, or deferred to the future, others will not necessarily agree. Bring such concerns into the open to manage and minimize what could otherwise become latent antagonism. Seek ways to draw colleagues in, and avoid isolating yourself and your project team. Be aware of and manage the potential problems caused by being the "new technology privileged."

Seek ways to draw colleagues in, and avoid isolating yourself and your project team.

The benefits to you for championing OOT in your company can be substantial: visibility, a reputation for leadership, and meeting and working with colleagues and senior managers otherwise unavailable to most IS professionals. You will meet and learn from the people who are making object technology into a burgeoning industry, and even have the opportunity to write articles and give presentations to conferences, should that be your interest. All these factors enhance your value and reputation, and bring credit to your company and to yourself.

Championing OOT will make you more valuable to yourself and to your company, and will demonstrate to others that you know how to avoid a crash and keep your bobsled on the track! The challenge for you will be taking control of the sled. ≡≡

≡ Your Turn ≡

How about it? How are we doing? This is the place to send us your message. Tell us what you like and what you don't, what you'd like to see but don't — or whatever other feedback you'd like to send. Send comments to Robert Shelton, 1850 Union Street, Suite 1584, San Fransisco, CA 94123; Fax: (415)928-3036; or, if you're ordering, send along with your order.

Please check whichever box applies:

	Read entirely	Scanned	Found helpful	Will refer to	Didn't read
Feature (*Douglas*)	❑	❑	❑	❑	❑
Feature (*Plant*)	❑	❑	❑	❑	❑
Interview	❑	❑	❑	❑	❑
Public Policy	❑	❑	❑	❑	❑
FYI	❑	❑	❑	❑	❑
Product Announcements	❑	❑	❑	❑	❑
Calendar	❑	❑	❑	❑	❑

Please rate the following **1** (dislike or strongly disagree) to **10** (like or strongly agree):

Presentation of material __ Comments: _____

Accuracy of material __ Comments: _____

Overall helpfulness in your job __ Comments: _____

What else would you like to see in THE HOTLINE? _____

Any other comments? _____

_____

_____

Name, title & phone (optional): _____

hotline on OBJECT-ORIENTED *technology*

❑ **Yes,** plug me into the latest thinking and developments in object-oriented technology. Enter me as a subscriber at the term marked below and rush me the current issue. This is a risk-free offer — I may cancel my subscription at any time and promptly receive a refund for the unused portion.

1 year (12 issues) ❑ $249

2 years (24 issues) ❑ $478 *(save $20)*

(outside US add $30 per year for air service)

Back issues @ $25 each *($27.50 foreign)*:

Vol.1, Nos. _____

Vol.2, Nos. _____ Vol.3, Nos. _____

❑ **Phone order**
Call Subscriber Services
(212)274-0640

❑ **Fax order**
Fill out term and name/address information, then fax to:
(212)274-0646

❑ **Send me a copy of**
THE INTERNATIONAL OOP DIRECTORY
@ $69 ($81 foreign)
(all directory orders must be prepaid; fax or mail credit card information to SIGS, 588 Broadway, #604, NYC 10012; make check payable to OOP Directory. Foreign orders must be prepaid in US dollars drawn on US bank.)

❑ **Bill me**
❑ **Check enclosed**
❑ **Credit card orders**
❑ MasterCard
❑ Visa

Card# Expiration Date

Signature

Name

Company

Street/Building#

City/Province

ST/Zip/Country

Telephone

Mail (and make out check, if applicable) to:
THE HOTLINE
Subscriber Services
P.O. Box 3000, Dept. HOT
Denville, NJ 07834
(foreign orders must be prepaid in US dollars drawn on a US bank)

Send me a complimentary copy of your related publication(s):

❑ THE JOURNAL OF OBJECT-ORIENTED PROGRAMMING

❑ OBJECT MAGAZINE

❑ THE C++ REPORT

Call Subscriber Services
(212)274-0640
or send order to
SIGS, 588 Broadway, #604
New York, NY 10012

D2EC

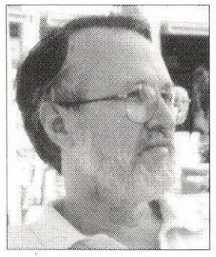
hotline on OBJECT-ORIENTED *technology*

VOLUME 3, NUMBER 7 THE MANAGER'S SOURCE FOR TRENDS, ISSUES & STRATEGIES MAY 1992

Implementing change

To renew, when we are deadlocked with the enemy, means that without changing our circumstance we change our spirit and win through a different technique.

— Miyamoto Musashi



Frank Douglas

Trial by combat is barbaric. It is fortunate that our culture has replaced the sword with the statistic and the lance with the fact, for trial by combat is today's accepted method for dealing with the complexities produced by broad-based, discontinuous change. Advocates of object-oriented concepts and technologies need to understand this, and accept the fact that introduction of O-O to a commercial enterprise will provoke a battle of words and ideas and, sooner or later, a confrontation with adherents to other approaches.

The value contributed by applications growing from an O-O conception will have to be proven to a corporate environment growing numb from too much novelty and rapidly becoming unresponsive to the imperative to change. At one time it was thought that all change represented improvement. Today, a fair chance to prove the usefulness of a new technology in a commercial environment may be obtained only with difficulty.

Object-oriented systems must share the stage with a variety of other approaches, technologies, and conceptual frameworks. These are not necessarily competing technologies. For example, there is no conflict at all between the fields of information visualization and object technology. Distributed processing works very well with object-oriented languages. Indeed, one has to work very hard to find a conflict between two generations of languages, such as Smalltalk and COBOL, when one carefully considers the proper domain of application of each language. New and older technologies can be integrated.

The profusion of change has actually created a situation where opportunities to integrate new technologies abound. For example, Smalltalk could readily implement a business information visualization application. This application could supplement a COBOL-based transaction data system. The new technology could demonstrate the added value it brings, and the overall position of the information system would be enhanced.

continued on page 4 ...

IN THIS ISSUE

FEATURE — Implementing Change	*Frank Douglas*	1	2	FROM THE EDITOR
FEATURE —What large-scale IT users want from O-O suppliers	*Norman Plant*	6	24	PRODUCT ANNOUNCEMENTS
INTERVIEW — Enterprise reengineering with CASE and objects	*Natasha Krol*	15	26	NEW PARTNERINGS & ACQUISITIONS
PUBLIC POLICY — Objects, security, and privacy on the electronic frontier	*David Bellin*	18	27	BOOK WATCH
FYI — Industry publication excerpts		21	27	CALENDAR

From the Editor



Survival in a slow economy among tough competitors demands flexibility. Organizations that can adapt may survive. Those which defer change choose their own fate, leaving only a question of when, not whether, they will decline. Yet our infrastructures are not built to support change, and reflect their designers. We have found the enemy. It is ourselves.

Take some curious examples. In a recent call to IBM, we requested help with an OS/2 problem. The install program was hanging. The obvious solution was to request help. After all, Jim Cavenaugh of IBM was telling attendees at Borland's International Development Conference '92 that IBM would support OS/2 on non-IBM hardware to compete against Microsoft for dominance of the desktop market. No problem, right? Wrong! Big Blue—yes, the company that is trying to get closer to their customers and be more responsive to the market—told us point blank "Since you are not running IBM hardware, we will not help you. Other manufacturers claim that their machines are IBM compatible. That is their claim. We do not test our products on other vendor equipment. We only guarantee IBM software to run on IBM hardware." Their representative didn't even bother to open with "I'm sorry"—I guess he wasn't! That's odd, though. Microsoft has helped us on many occasions with questions about DOS and Windows configuration—and we don't run Microsoft hardware! Who's changing?

Then there is Informix—an open systems database vendor that is making a major foray into the object technology market. As Informix customers are well aware, this database vendor has a history of rough customer service and inconsistent customer data due to separate databases in each department that deals with customers. It seems that renewal bills for technical support contracts don't always reach the customer. Presumably, the unfortunate customer would discover the problem upon calling for support, send in their renewal, and all would be happy. There's only one gotcha: when you don't receive a bill, your support contract expires. When the contract expires, you the happy customer are slapped with a 150% penalty to renew your support contract—and you cannot receive new software releases without a support contract. Of course, the now not-so-happy customer is welcome to "plead your case with one of the supervisors to make a change in the policy." Or, in this world of client-server architecture and database independence, the customers can do the changing themselves—change vendors, that is!

And customers are doing the changing, even when their infrastructure would trap them in proprietary environments and inflexible technology—provided that their information technologists can help them adapt. And here, in the seemingly innocent realm of customer service, are many of the central issues facing organizations interested in object technology (OT): flexibility, freedom to choose the best solution for the business, portability and transparency across platforms at multiple layers (hardware, operating systems, networking, and database), product interoperability, product reliability, vendor support, and changing human attitudes.

Most of us face enough challenges and difficulties with these issues in our own organizations that vendors and technologies that don't support our needs are impediments, not assets. Realizing that our vendors face the same challenges as we ourselves experience helps us little, as many of us are seeking from our vendors solutions, not mirror images of our own troubles as we struggle to change our own people.

It is this dilemma of change that our feature author Frank Douglas addresses. As technologists, our strong suit is technology; as change agents, our strength should be people skills. Perhaps we now have an insight into our vendors' change problems—they too are more facile with technology than with people (their customers or their own)! Yet, as Mr. Douglas ob-

Editor
Robert Shelton

SIGS Publications, Inc. Advisory Board
Tom Atwood, *Object Design*
Grady Booch, *Rational*
George Bosworth, *Digitalk*
Brad Cox, *Information Age Consulting*
Chuck Duff, *The Whitewater Group*
Adele Goldberg, *ParcPlace Systems*
R. Jordan Kreindler, *General Electric*
Meilir Page-Jones, *Wayland Systems*
Tom Love, *Consultant*
Bertrand Meyer, *Interactive Software Engineering*
Sesha Pratap, *CenterLine Software*
P. Michael Seashols, *Versant Object Technology*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

Hotline Editorial Board
Jim Anderson, *Digitalk, Inc.*
Larry Constantine, *Consultant*
Mary E.S. Loomis, *Versant Object Technology*
Reed Phillips, *Knowledge Systems Corp.*
Trygve Reenskaug, *Taskon A/S*
Zack Urlocker, *Borland International*
Steven Weiss, *Wayland Systems*

SIGS Publications, Inc.
Richard P. Friedman, Founder & Group Publisher

Art/Production
Kristina Joukhadar, Managing Editor
Pilgrim Road, Ltd., Creative Direction
Elizabeth A. Upp, Production Editor
Jennifer Englander, Art/Production Coordinator

Circulation
Diane Badway, Circulation Business Manager
Kathleen Canning, Fulfillment Manager
Vicki Monck, Circulation Assistant
John Schreiber, Circulation Assistant

Marketing
Sarah Hamilton, Promotions Manager
Caren Polner, Promotions Graphic Artist

Administration
David Chatterpaul, Bookkeeper
Frank Ferrantelli, Bookkeeper
Claire Johnston, Conference Manager
Cindy Roppel, Conference Coordinator
Jennifer Fischer, Public Relations
Lauren Willinger, Administrative Assistant

Margherita R. Monck, General Manager

Jane M. Grau, Contributing Editor

hotline on OBJECT-ORIENTED *technology*

The Hotline on Object-Oriented Technology (ISSN #1044-4319) is published monthly by SIGS Publications, Inc., 588 Broadway, NY, NY 10012, (212)274-0640. © Copyright 1992 SIGS Publications, Inc. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the U.S. Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher. Mailed First Class. Subscription rate — one year (12 issues) $249, Foreign and Canada $279. Single copy $25.

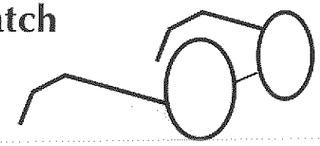
POSTMASTER: Send address changes & subscription orders to The Hotline, Subscriber Services, P.O. Box 3000, Dept HOT, Denville, NJ 07834.

Submit editorial correspondence to Robert Shelton, 1850 Union Street, Suite 1548, San Francisco, CA 94123. Voice: (415) 928-5842; fax: (415) 928-3036.

SIGS PUBLICATIONS

Publishers of *Hotline on Object-Oriented Technology, Journal of Object-Oriented Programming, Object Magazine, The X Journal, The C++ Report, The Smalltalk Report,* and *The International OOP Directory.*

Book Watch



The "Book Watch" column does not contain book reviews. These listings are abstacted from press releases provided by the publishers, and no endorsement is implied. Please send announcements to the Editor: Robert Shelton, 1850 Union Street, Suite 1584, San Fransisco, CA 94123; Fax: (415) 928-3036.

Prentice Hall, Simon & Schuster Education Group, Professional Technical & Reference Publishing, 113 Sylvan Ave., Rte. 9W, Englewood Cliffs, NJ 07632 announced the publication of several titles on object technology:

Object-Oriented Analysis and Design by James Martin and James Odell addresses the difficulties encountered in founding analysis techniques on a particular technology. Written for the typical systems analyst, the book teaches both the underlying philosophy of object orientation and its practical usage.

Class Construction in C and C++: Object Programming Fundamentals, by Roger Sessions, is a complete, in-depth introduction to object-oriented programming, developing classes, and C++ for C programmers.

Object-Oriented Programming with C++ and OSF/Motif, by Douglas A. Young, takes the object-oriented approach to designing and building applications for interactive window-based user interfaces using C++ and the OSF/Motif user interface toolkit.

A Book of Object-Oriented Knowledge: Object-Oriented Design and Implementation, by Brian Henderson-Sellers, is an introduction to object-oriented analysis and design rather than programming techniques.

Modula-3, by Samuel P. Harbison, PhD, provides programmers with a clear and complete description of the new Modula-3 programming language and shows them how to write clear and correct code.

Distributed Object-Oriented Systems Analysis, by Prabhat K. Andleigh and Michael R. Gretzinger, presents and evaluates application design and development methodologies. The authors have developed a unique Frame-Object Analysis Methodology for advanced modeling.

The C++ Programmer's Handbook by Paul J. Lucas is a quick reference guide available for AT&T's Release 3.0 of C++. The book is a programmer's reference for fast look-up of information on C++, the preprocessor, the input/output library, and other libraries for both novice and experienced C++ users.

Tools 5: Technology of Object-Oriented Languages and Systems by Tim Korson, Vijay Vaishnavi, and Bertrand Meyer, contains the proceedings of the Tools 5 conference held at the University of California, Santa Barbara in the summer of 1991. Topics covered include object-oriented databases, reusable objects, a comparison of object-oriented languages, object-oriented user interfaces, and object-oriented project management.

C++ for Programmers, Book/Disk Version by Sharam Hekmatpour now includes a disk containing all in-text programs.

Calendar

The Hotline Calendar presents conferences and meetings that focus exclusively on object-oriented technology. To have a meeting or conference listed, please send the dates, conference name and location, sponsor(s) and contact name and telephone number to the Editor: Robert Shelton, 1850 Union Street, Suite 1584, San Francisco, CA 94123; fax: (415) 928-3036.

May 18–22	June 1–4, 1992	June 8–12, 1992	June 16–18, 1992	July 14–17, 1992
InterOp	Object Expo	USENIX	SD '92	Object Expo Europe
Washington, DC	New York City	San Antonio, TX	London, UK	London, UK
Contact: 415.941.3399	Contact: 212.274.0640	Contact: 614.588.8649	Contact: 081.742.2828	Contact: 212.274.0640

Partnerings & Acquisitions

Object Design, Inc. announced it has established a worldwide network of top distributors for its ODBMS, ObjectStore. Object Design's international partners are: **Apscore International Pty. Ltd**, headquartered in Neutral Bay, Australia; **DyM** based in Madrid, Spain; **Elsa Software** located in Cedrex, France; **ENEA DATA AB**, headquartered in Malmoe, Sweden; **Patzschke + Rasp**, located in Wiesbaden, Germany; **Sema Software Technology**, based in Chippenham, England; and **Toyo Information Systems Co, Ltd**, headquartered in Tokyo, Japan.

Object Design also announced it has closed a $2.5 million round of financing led by Los Angeles-based **Brentwood Associates** and including its existing investors. The investment represents a 30% increase in Object Design's per-share valuation over equity placed during the company's third round of financing seven months ago.

ONTOS, Inc. announced a cooperative marketing and product integration agreement with **ParcPlace Systems**. Under the terms of the agreement, the two companies will cooperatively integrate and market ONTOS DB, ONTOS Studio, and Objectworks/C++. The product integration is avilable now and is currently supported on the Sun SPARCstation.

ONTOS also announced a coooperative marketing and product integration agrement with **CenterLine Software, Inc**. The newest release of the ONTOS DB object database, Release 2.2, and ONTOS Studio are now integrated with the ObjectCenter programming environment for C++ from Centerline. The product integration is currently supported on the Sun-4 SPARC.

Petrotechnical Open Software Corporation (POSC), **Corstar Business Computing Company, Ltd.**, and its joint venture partner **N.V. Electriciteitsbedrijf Suid-Holland** (EZH) have joined the **OMG** as corporate members. The OMG also announced that over 35 companies plan to support the Common Object Request Broker Architecture (COBRA) specification adopted by the OMG last fall.

Lucid, Inc. announced the appointment of Simon Clephan as manager, international distribution. Clephan will manage international distribution for Lucid's family of C and C++ compilers and the Energize Programming System focusing first on Europe, Japan, and Australia. Clephan was most recently product marketing manager at IntelliCorp, Inc.

Lucid, Inc. also signed a technology agreement with **HaL Computer Systems, Inc.** to provide Lucid's SPARC-based C, C++, and Fortran compiler technology to help build high performance and speed into HaL's compilers for its family of SPARC-based computer systems. Hal Computer Systems' computer systems will be targeted at commercial users of open systems. Lucid will provide technical support and assistance to HaL's development requirements as required.

MetaWare, Inc. announced an agreement with **Phar Lap Software** to license MetaWare's Globally Optimizing 32-bit High C/C++ Compiler and Phar Lap's 386/DOS-Extender together in a package offered by each company. This agreement enhances an already successful five-year business and marketing relationship between the two firms.

UNIX System Laboratories, Inc (USL) and **Hewlett-Packard Company** (HP) announced that HP will provide exception- and event-handling technology for USL's C++ language system. This announcement grows out of cooperation between HP and USL throughout HP's development of this exception handling feature.

Object Technology International, Inc. announced that products developed under the recently announced **IBM** Applications Business System/**OTI** development agreement will include remote database access conforming to IBM's distributed relational database architecture (DRDA). Initially developing a DRDA applciations requester, the IBM-OTI development agreement is intended to enhance the capabilites of AS/400 programmiers using graphical user interfaces, cooperative programming and object-oriented programming.

serves, the wrong approach to change undermines the organization's willingness to change, reduces flexibility, and increases resistance to new ideas. And the persistent demand for change can wear down those whose mission it to be skilled in the technically complex. Ironically, this intense change is increasing the technological opportunities as it breeds its own resistance.

Learning the new requires a willingness to become "not skilled" on the way to developing new skills, and few organizations support this learning cycle effectively. Most, as Mr. Douglas points out, address new ideas through trial by combat—under circumstances that reflect more on the political power of the combatants than on the purpose of the technology. Our strengths with technology—insight, ability to think abstractly, comfort with complexity, singularity of purpose—which would seem to work in favor of flexibility can as easily work to our detriment in a change-demanding environment.

Numbness to change in IT divisions is not simply an inherent, human-change resistance phenomenon. Marketing hype and silver bullet technologies have unrealistically inflated expectations and derailed enough well-meaning practitioners that change for the sake of a new technology is decreasingly appealing. Perhaps this aspect of *la resistance* is maturity and realism. Technology has been a value in and of itself in our culture, and we are only now rediscovering the original purpose of information technology as a support tool to increase business effectiveness. That phrases like "business value of technology" abound suggests a realignment of values that will be reinforced by the trend toward useful new technology being driven by our business partners rather than by ourselves.

But resistance is resistance and, as Mr. Douglas states, "a fair chance to prove the usefulness of a new technology in a commercial environment may be obtained only with difficulty."

Norman Plant of the Object Interest Group in England joins us again (see HOTLINE, March 1992) with a substantial statement on the needs of large IT organizations and points out critical areas in which this emerging technology should—and should not—be deployed. Mr. Douglas observes that new technology is often trialed in inappropriate domains. Mr. Plant clearly identifies those domains from the perspective of organizations with tens of thousands of online workstations and millions of daily transactions. He also proposes areas of strength for OT that allow integration with the areas of strength (or at least stability) of our legacy systems.

An observation: those who are saddled with the cost and burden of legacy systems are looking for ways to grow new capabilities with OT, and view outright replacement of those core systems as beyond the reach of either their organization's finances or a relatively new technology that was not designed to solve transaction-processing problems on that scale. Risk containment is the term Mr. Plant uses. Those not faced with 20+ year old, 40,000 online user systems seem to view outright replacement more as an opportunity than as a risk. Forced to choose between an inflexible but well-tested core system that correctly handles passenger ticketing and a new core system that offers flexibility and a tool-kit approach to solving problems—well, you choose! When your entire revenue steam is on the line, technology becomes but an amusing sideshow.

This is not to suggest that business reengineering and the replacement of archaic core systems cannot happen. The business reality facing air carriers like British Airways, United, and American Airlines should serve as a sobering reminder to those of us who are already converts to OT. Turning back to our feature article, the converts have to prove the business value of OT, and the best proving ground at this point in the maturity of OT is often not the core. To paraphrase Mr. Plant, applications that extend the reach and range of the business by augmenting the core offer the greatest return on investment for the application of OT today.

Dr. David Bellin addresses a different aspect of adaptation—the unpreparedness of our legal and policy infrastructures for this new information technology, and specifically for some of the implications of OT. Consider the licensing issues, security risks, and product liability associated with extensive class library use. How we validate libraries—especially those for which we will never see source—is an issue that few have addressed on par with the obvious benefits of building applications from libraries of software-ICs, to use Brad Cox's term. These issues are critical to large organizations, like Mr. Plant's British Airways, which would start using OT for reach and range applications then migrate it over time (at some point after accepted technology maturity) into core applications.

Yet other new exposures are created by the electronic business environment that OT enables. Because of both heritage and design, object-oriented development environments, "objectbase" management systems (OBMS) and even graphical user interface (GUI) builders encourage application client-server separation. While client-server architecture within a single machine facilitates portability and interoperability, the real power is seen only across networks where heterogeneous client and server hosts cooperate across local or global networks. New issues of security, privacy, need-to-know, and respect for the expression of ideas in conflict with one's own are unavoidable in this networked electronic environment.

Global networks confound the problem by adding international politics, different cultural values and legal systems, and asymmetrical views on expression, belief, ownership, and privacy. As Dr. Bellin expresses, these are issues to be taken seriously. Even within a single nation—the USA—what would be established policy with regard to physical property or verbal communications is not honored where electronic networks are concerned. Accepted and assumed protections are not enforced. The world of electronic reality raises new problems for a legal culture whose roots may be found in physical property law.

In our interview with industry analyst Natasha Krol, we address cultural change from a different perspective: business process reengineering. The real opportunities offered by OT are, as Ms. Krol points out, to be found in the compelling nature of the underlying paradigm itself and its impact on how we can think about business and systems. Effective use of OT requires that we thoroughly understand the atomic fundamentals of the business our applications support, and the paradigm provides support for that return to first principles. The promise of OT is in making cleaner the translation of business into applications.

But where does one start with OT? Ms. Krol suggests that IT apply OT to model and understand the legacy systems, thereby gaining experience with OT and a better understanding of the core that will be augmented or—eventually—replaced. As information technologists better understand both OT and the businesses they support, Ms Krol suggests, they can more effectively partner with the business community. Within that partnership, avenues can be found for leveraging the business with useful information technology.

Our authors this month strike some common themes about change from notably different perspectives:

- we information technologists face the greatest problem with change
- our business partners are ahead of us, and don't count on them good-naturedly waiting for IT to catch up
- change will continue to be the norm

That OT requires so substantial a perspective shift from the computer-centricity of batch processing applications is one of our greatest challenges as information technologists. With 80% of IT resources committed to maintaining legacy systems built a decade or more ago, it's not surprising that the tool-kit view of flexible, user-empowering applications is a stretch! This is, however, a stretch that we have to make to remain useful to our organizations and customers. It is also a goal that I believe we can reach, given the personal commitment to do so. ≡≡

IMPLEMENTING CHANGE *continued from page 1*

Life is never so simple. People are involved, as is knowledge. Every decision made will be made by someone who has some type of self-interest. Each individual will also have an incomplete idea of the purpose and capacities of the new technology and of the problem at hand. The advocates and practitioners of each competing approach will not cede their positions. Indeed, the "assembly of facts" will rapidly become a set of fairly standard assertions:

- The "new technology" is not new. "We did that in assembler when I started in this business."
- Existing technology can do that just as well as the new technology.
- The existing technology is a proven approach.
- The new technology will produce a significant increase in productivity and reduce maintenance cost.

It should also be noted that an opportunity to introduce a new technology is also context bound. It will be created by the business environment. One of the changes that has been reaching a level of critical importance is emphasis on the business relevance of every technical change made to a corporate technical infrastructure. Purely technological changes will probably be thrust aside. However, innovations that address the business's competitive stance or its product marketability will have a greater opportunity to demonstrate value.

A contest is the inevitable result of this combination of partial understanding, public interest, and continual change. The contest is a duel between two technical approaches to a business problem in which winning or losing becomes the major issue rather than the objective issues surrounding the change: the cost, benefit and potential offered by the new technology. It is also unlikely that the contest will take the form of a competition between the two technologies. It is more likely that the battle will be settled by giving the new technology a "chance to solve an important business problem." All too often, this chance is the end result of considerable maneuvering by all parties.

Frequently, the outcome of the contest is settled by the actual assignment made. At one company, the advocate of Smalltalk was maneuvered into taking on a project to construct a transaction-based system capable of outperforming a COBOL-CICS system that benchmarked at 300 transactions per second. Of course, it is possible that the advocate could design and build a system that does just that, and introduce an entirely new way to build transaction systems. But it is more likely that COBOL-CICS is perfectly suited to what it was doing, while Smalltalk is better suited for other things.

It is unlikely that any advocate of a new technology would get maneuvered into such an absurd test without the help of a dedicated adversary. If left alone, the advocate can develop a situation that provides a significant business benefit and demonstrates the strength of the technology. The situation is complicated by the participation of interested parties from both the technical and nontechnical areas of the business. All these participants have either embraced or rejected the new idea. Those who have rejected the idea will generally manifest their opposition in the "contest." However, those who have accepted the idea with the greatest fervor frequently become a major source of problems during the maneuvering period of the confrontation.

Conversion to the new idea is the foundation of change. Sharing the vision with others is essential. The advocate can provoke an unfavorable contest. Technologists are notorious for their optimism. Every IS development manager has been stung one or more times by someone overpromising and then underdelivering. In the words above, "We did that in assembler when I started in this business." It is much more common in newer technologies, however, when practical limitations seem more like the carping of opponents than real barriers.

Nontechnologists are increasingly becoming advocates for new technologies. It has become quite common for the marketing department or the Chief Financial Officer to advocate shifting to object technology. The dynamics leading to a "test" can originate from nontechnological sources more readily than from technological ones. After all, these individuals are conducting the business. They have an immediate interest in improving anything that will help them do so. A contest, not unlike the great air races that helped to prove aviation, frequently seems to be the best way to decide on the suitability of a potentially more helpful way to conduct business. Confrontations and tests often are the only way new technologies can make their way into the business culture.

Even though the contest is supposed to be won or lost based on *suitability for business*, the advocate is often hampered when negotiating the subject matter of the test. The precontest of determining the subject matter cannot be won "at all costs." Advocates of the new technology are a factor that must also be considered. If the maneuvering is not handled correctly, the test may provide fuel for a "breakaway" group that will be dedicated to implementing the new technology "correctly." Implementation of CASE and relational databases was frequently complicated by these breakaway groups.

After the subject of the test has been decided, events will move into a new phase. The battle will escalate and the number of issues will proliferate wildly. This phase will expose the "soft underbelly" of the ideas to circumstances intended to demonstrate its suitability for pragmatic use. In this phase, the more muscular primary com-

Comeau C++ 3.0 With Templates

Comeau Computing
91-34 120th St.
Richmond Hill, NY 11418
718.945.0009

Comeau Computing announced the availability of **Comeau C++ 3.0 With Templates**, its multiplatformed C++. The new release includes support of ANSI C++ templates, which enhances the code reuse capabilities of C++. Comeau Computing also announced its intent to provide support for numerous back-end C compilers and additional ports of Comeau C++ 3.0. Comeau C++ is especially designed with cross-platform and cross-operating system capability. Previous versions of Comeau C++ supported MS-DOS, all UNIX 386's, SCO UNIX 386, SCO XENIX 386, the AT&T 382, AmigaDOS, AT&T UNIX PC 7300 and 3B1, the Sun-3, and the IBM RS/6000 A1X. Comeau C++ 3.0 adds support for the Sun SPARC, and others to include the Atari ST, UNIX SVR4, MS-Windows, and OS/2. .

ENVY/Developer R1.30

Object Technology International, Inc.
1785 Woodward Drive
Ottowa, Ontario, Canada
K2C 0P9

Object Technology International, Inc. announced the immediate availability of its object-oriented product development environment, **ENVY/Developer R1.30** for Smalltalk/V PM V1.3 and Smalltalk/V Windows V1.1. This product provides a powerful concurrent software engineering environment for systems and applications development. Team support, version control, and configuration management are seamlessly integrated with Smalltalk/V's programming environment. Release 1.30 includes ENVY/Packager, OTI's tool for delivering small, stand-alone Smalltalk applications. With ENVY/Developer, development teams using Smalltalk may work concurrently on both OS/2 and Windows. Sophisticated change browsers support the ability to compare different component versions.

Action!

Expertelligence, Inc.
5638 Hollister Ave., Ste. 302
Goleta, CA 93117
805.967.1797

ExperTelligence announced a new version of its intellegent interactive graphical application development system **Action!** for Version 1.3 of Digitalk's Smalltalk/V PM. Action! V1.3 adds new CUA 91 objects like the Spin Button and the Slider. It also gives more control to the developers with the powerful Properties Editor and Coordinate Windows. It also allows interface compatilibity with ExperTelligence's Macintosh Lisp versions of Action!. Action! is totally object-oriented and dynamic, and is also totally machine, system and language independent.

GUI Master 2.0 Professional

Vleermuis Software Research
P.O. Box 2584
3500 GN Utrecht
The Netherlands
31.30.32.49.44

GUI Master 2.0 Professional, the latest release of the application framework for development of GUI applications for OS/2 and Windows, is now supplied with an Application builder that enables users to draw their application on the screen. Online help and complete CUA help are also available. Features also include a C++ Browser, which lets developers browse through the source code of all GUI Master classes giving them total programming control. GUI Master 2.0 Professional can be installed on either an OS/2 or a Windows platform.

Style

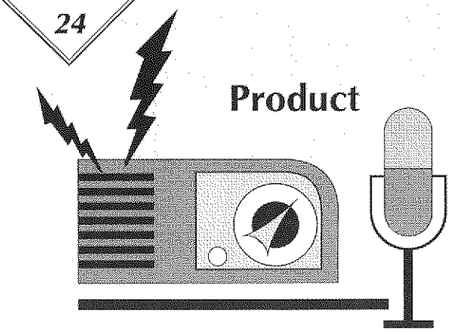
Software Ingenuities, Inc.
P.O. Box 1586
Ballwin, MO 63022
314.391.7772

The Windows version for **Style** is now available from Software Ingenuities, Inc. Style is an all-purpose class library for object management. It can help C++ programmers develop faster, smaller, and more maintainable applications. With a single call, programmers can perform operations such as: access distantly linked objects, collect objects meeting certain criteria, destroy selected objects, save and load objects from disk files, print selected objects, print links between objects, and search for a specific object.

ProVision:Windows Version 1.01

SofDesign International, Inc.
1303 Columbia Dr.,
Ste. #209
Richardson, TX 75081
214.644.0098

SofDesign International, Inc is now shipping **ProVision:Windows Version 1.01**, a completely object-oriented human interface development library for Clipper 5.01. ProVision:Windows provides Clipper programmers with features necessary to build true virtual-paged, modeless, event-driven human interfaces. It handles all windowing and mouse selection details. ProVision:Windows supports dialogue, box windows, message box windows, icons, and Multiple Document Interface (MDI) windows. The library includes a tightly integrated mouse subsystem. Windows, controls and menus all support the mouse automatically requiring no programmer intervention.



Product Announcements

Product Announcements is a service to the readers of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY; it is neither a recommendation nor an endorsement of any product discussed.

HP C++ 3.0

Hewlett Packard
PR Dept. 3000 Hanover St.
Palo Alto, CA 94304
415.508.6600 ext.7717

Hewlett Packard Co. introduced **HP C++ 3.0**, an object-oriented programming language for application developers working on HP's Apollo 9000 workstations and HP 9000 business systems and servers. HP's implementation of exception handling conforms to the ANSI C++ committee's definition. HP's fully integrated C++ tool set is a key component in the development of applications for a wide range of markets, including computer-aided design, telecommunication, user-interface development, business graphics, database development, and multimedia.

ONTOS Studio

ONTOS, Inc.
Three Burlington Woods
Burlington, MA 01803
617.272.7110

ONTOS announced **ONTOS Studio**, an interactive, front-end development tool that allows the ONTOS object database developer to build graphical user interfaces for a range of applications and store these user interface (UI) components directly in the database. The product is currently available for Sun SPARC workstations. In a related announcement, ONTOS also introduced Release 2.2 of its ONTOS DB object database with expanded support for workgroup applications, increased support for integrated object tools and languages, new platform support, as well as performance tuning options.

Archetype Document Engine

Archetype, Inc.
100 Fifth Ave.
Waltham, MA 02154
617.890.7544

Archetype, Inc. introduced the **Archetype Document Engine**, an object-oriented development environment for NeXT Computers that enables developers and users to quickly build modular, expandable publishing applications. The Archetype Document Engine provides a tool kit of ready-to-use objects that can be dynamically linked to form fully featured publishing applications. DE applications can provide multiuser access to a single document, including multiple users working on different parts of the same page. The DE provides a locking mechanism for objects akin to file locking in a network operating system.

OTProvide

Objective Technologies, Inc.
7 Dey St., Ste. 1502
New York, NY 10007
212.227.6767

Objective Technologies, Inc. announced **OTProvide**, a database link and query manager between relational databases and Lotus' Improv. OTProvide allows users to easily access data stored in relational databases and integrate that information into Improv models without a Structured Query Language. The product provides point-and-click query generation that lets end-users who do not know SQL to quickly and easily access information stored in databases. OTProvide runs on NeXT 2.0 system software, Improv with the API Kit, and Sybase. OTProvide version 1.0 provides access to Sybase databases only. Version 2.0, due to be released concurrently with NeXt Inc.'s DBKit, will provide access to Oracle databases.

SoundBursts Digital Sound Collections

MIDIapolis Systems
3208 W. Lake St., Ste. 133
Minneapolis, MN 55416
612.822.1604

MIDIapolis has begun shippng **SoundBursts Digital Sound Collections** for NeXTstep computers. SoundBursts allows users to enhance their NeXTstep presentation, musical compositions, voice annotations and applications with the impact of CD-quality sound. Two SoundBursts collections are available—Sound Effects and Percussion. Each collection includes 12 diskettes (an average of 125 soundfiles per collection) of profesionally recorded sounds.

View.h++, MouseWrapper.h++

Rogue Wave Software,Inc.
P.O. Box 2328
Corvalis, OR 97339
503.754.3010

Rogue Wave Software, Inc. announced two new C++ libraries: **View.h++**, a commercial library based on Motif, and **MouseWrapper.h++**, a system for simplifying interactive graphics. View.h++ for Motif offers two levels of functionality. At its lowest level, it encapsulates 100% of the functionality of the Open Software Foundation's industry standard Motif Graphical User Interface in a C++ library. At the higher level of functionality, View.h++ for Motif offers a Model-View-Controller architecture with an even more abstract and powerful programming interface. MouseWrapper.h++ is a system for managing object picks, drags, rubber banding, and interactive graphics. Grabbers are also included for interactive construction of complex graphical composite objects.

Send Product Announcements to Robert Shelton, 1850 Union Street, Suite 1548, San Francisco, CA 94123, fax: (415) 928-3036. Include company name, address, and phone number.

ponents will often be compromised or even displaced by the functions supporting the primary ones.

For example, one strength of an object-oriented development environment is that developers can become extremely productive. Changes can be incorporated easily; the business world can find a direct counterpart in the class libraries and in the programming manipulations of developers. However, the time to learn the environment is very long, and productivity during that time is well below that of third-generation developers. Changes are incorporated easily in a single-image system. Once multiple developers begin to work independently, the process of configuration management becomes an accurate simulation of the Tower of Babel.

In this phase, practical shortcomings will surface unexpectedly and repeatedly. This is often surprising to the advocate, but it shouldn't be. Nothing is perfect about a new technology. The products that implement it are immature; everyone should expect that the product will be unable to handle a great many practical issues. The technology and the techniques for its use need to accommodate all types of situations that are inconsistent with the main premise of the technology. There will be many compromises with the pristine view the technology afforded at the beginning of the contest. These changes offer new opportunities for the contest to be lost.

Arguments about practical issues will become dominant during the testing phase. This cannot be avoided; opponents will see that every problem is highlighted. A truly dedicated opponent will attempt to raise each issue as being of critical importance. Resolving the substantive issues successfully is important, of course. There must be training; there must be configuration management; there must be an appropriate class created for some unnamed business problem. But, answering the secondary issues created by the rhetoric will become essential to the final outcome of the test. The secondary issues represent those issues having nothing to do with the technology, per se, but that every decision maker understands. They are frequently presented as "integration" issues, although they seldom represent integration.

Secondary issues can run the gamut from paper-based approval methods to security. In some cases, the secondary issues can lead to decisions that will virtually incapacitate the product being used to test the technology. An apt case in point was provided by one company testing DB2. During the design of the test, IS management imposed the requirement that all DB2 database administration be conducted in batch through a secure scheduler. This is extremely difficult, so the test failed when the DBAs were unable to maintain the tables during the test. The decisions made on these issues will determine the success of the test. The advocate of the new technology must be alert to the issues the moment they emerge, and must understand the forces that have brought them forth. Addressing these issues will normally require the most dramatic changes in the use of the new technology.

Whatever specific manifestation of the new technology is chosen for implementation during the test, the technology it represents will have its central tenets modified. This is a source of new problems for the emerging technology. The very people who have brought the technology to the edge of implementation may find themselves revolted by the compromises with the ideas they originally advocated. They may begin to insist on "keeping to the model." These insistences may make it very difficult to accommodate demands imposed by business realities. It is difficult to know when to remain true to theory, and when to bow to necessity. Former enthusiasts can be formidable adversaries if sufficiently disillusioned. On the other hand, theory is never adequate in and of itself. It must be expanded empirically. As with the other components of the test, the ability of the new technology to be responsive to the practical is on trial.

> ***Former enthusiasts can be formidable adversaries if sufficiently disillusioned.***

The battle has only begun when the test has reached a successful conclusion. Frequently, the test has only been a pilot. The results will almost certainly be dismissed as being inapplicable to the real world. There might even be an effort to prevent the new technology from repeating the results with a "real" application. What is increasingly prevalent is the culture shock spreading through American corporations. Technology is changing much faster than the technicians are comfortable changing themselves.

As one set of changes succeeds the next, a yearning for some type of stability is becoming increasingly prevalent. Members of organizations achieve that stability by translating new ideas, tasks, and possibilities into things that are familiar and clearly understood. It is no use pointing out that a "paradigm shift" will be required to properly employ the new technology because the very concept of a paradigm shift has been assimilated into a simpler idea of being smart enough to understand what's happening. Of course, everyone who is "smart enough" to understand a paradigm shift is certainly smart enough to understand a new form of technology.

It is here, in the wider sphere of acceptance, that any new technology must win its most decisive challenge. A

continued on page 14...

What large-scale IT users want from O-O suppliers

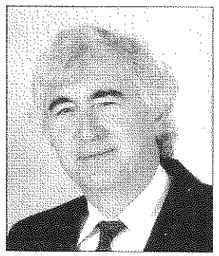
The Object Interest Group (OIG), consisting of 14 large UK companies, two Government departments, and one University, was formed in May, 1990.

Its purpose is to understand and assess object technology, get hard evidence to support the assessment, enable member companies to position the technology in their IT strategy, and to communicate to suppliers what large-scale users want in terms of method, tools, and support. Project members include Barclays Bank, British Petroleum Research, British Aerospace, British Airways, British Steel, British Telecom, Department of Trade and Industry, The Treasury, Ford Motor Company, Imperial Chemical Industries, Lloyds Bank, National Westminster Bank, Prudential Assurance, Rolls Royce, Royal Insurance, Trustee Savings Bank, and University of Strathclyde. No funds were involved. Each company agreed to contribute at least 40 man days; in total, some 1500 man days went into the assessment. The business case for sharing was return on investment: 40 days in, 1500 out.

In my previous article[1] I presented our overall assessment of the technology, which can be summed up by our shared conclusion:

> Object orientation is potentially one of the most powerful technologies ever to become available to the IT industry and its users. As such it demands high-calibre management. It is not a panacea but a high-power tool—dangerous if misused but capable of great things.

In this article, I shall focus on why O-O is not a panacea and, in particular, why it is too early for the wholesale migration of any large scale IT function entirely to O-O. As we saw both promise and insufficiency in state-of-the art O-O, one of the OIG's objectives was to identify to the supplier community what it is we want from O-O.



Norman Plant

We did this by trying to define what maturity would look like, assessing the current state of the art, and allowing the gap to represent our needs. One problem with this approach is that until you have a reasonable level of experience on real projects it is difficult to know what the real problems and hence what your real needs are. At the time of our assessment, February, 1991, none of our members had extensive experience with O-O, although this is now rapidly changing. Nevertheless what follows is our best attempt to state and gather all our needs together in one place. I appreciate that the products and services picture is continuously evolving, but what follows is a reasonable representation of what are still the concerns of the large IT division as regards O-O.

METHODOLOGY INTEGRATION

Many industries have made substantial investments in current methodologies such as the information engineering (IE) approach. Like many other approaches, though not uniformly successful, IE has delivered enough value that it will not simply be discarded by the IT divisions that have invested in it. Therefore, understanding how IE and O-O coexist will be a critical step in gaining acceptance for O-O in large IT divisions. This immediately raises a number of questions:

- What is best in the IE and O-O philosophies?
- To what extent can IE- and O-O-based systems coexist?
- What are the practical migration paths from IE to O-O, and which of these is the most practical route given available technology, the state of development of O-O methodologies, and our intended use of O-O?
- What changes to our working methods are required, how do we incorporate them, and how should we document these processes and methods?
- Change control will be a problem that will be further exacerbated by the introduction of workstations and client/server technologies: how do we manage this change?

Similar questions to those above arise for other current methodologies such as Yourdon and SSADM. If vendors of object technology fail to provide a migra-

Reuse

...The situation [of modules in object technology] is analogous to the use of integrated circuits (ICs) in electronics...The internal complexity of an IC module may be very great, but its connection to the outside world is limited to a few very well documented pins: it is neither possible nor desirable to connect directly to the insides of the device. Information hiding serves both to simplify the application of modules as well as to protect proprietary construction methods; this is analogous to the common practice in the packages software industry of selling only the executable binaries, not the source code...

The object is productivity, T.C. O'Haver, ***INTELLIGENT INSTRUMENTS & COMPUTERS****, 4/92*

...Software developers need to reuse more program code and need to use higher-level building blocks in order to cut the terribly long time it takes to develop modern applications. [Brad] Cox passionately believes that object-oriented programming and the concept of what he calls a software IC—the software equivalent of an integrated circuit—are the way to go. There is evidence to support his ideas. Borland, for example, keeps putting out major upgrades to its products, such as Quattro Pro and Objectvision, every six months. Borland claims that object-oriented programming is the key behind their accelerated release schedule...

Knowledge is good for you, Eric R. Johnson, ***PUGET SOUND COMPUTER USER****, 2/92*

Making the Change

...The only drawback to object-oriented technology is that, while not exactly new, relatively few IS professionals have had experience with it. Object-oriented program structures are so radically different from conventional programs that many older Cobol programmers—who are used to breaking down tasks and sub-tasks, etc.—have a lot of trouble learning object-oriented programming. (Conversely, recent college graduates without years of old techniques under their belts grasp it quickly.)...It is quite possibly, believe some observers, the foundation of the next generation of development tools. If it can surmount the shortage of expertise, it might well be.

Bank tech review—liberated data: free at last, free at last, ***ICP BANKING SOFTWARE REVIEW****, Winter/91*

...The fastest changeover to object-oriented technology is occurring in the software that engineers use to automate the design of complex electronic and mechanical systems. Traditionally, in computer-aided design, each engineer uses a set of specific software-based tools. Object-oriented frameworks will allow engineers to put together a suite of tools from different vendors. The object-oriented software will make it practical for each of these design tools to share a common graphical interface, making them easier to learn and to use. As designers work for hours on their section of an overall project, the object-oriented database will help keep tabs on any changes that might affect other sections. Moreover, object-oriented databases take some of the tedium out of designing. An electronics engineer, for example, could simply specify what task a circuit should perform; the object-oriented database will help translate this specification into a circuit design...The changeover in business data processing will come more slowly, [Object Design's Thomas] Atwood says, because present relational databases are well-suited to the file structures common to business applications Object technology will be used instead to integrate the computer's underlying operations....

Software's new, Robert Haavind, ***TECHNOLOGY REVIEW****, 3/92*

...[Gartner Group's David Stein]"Object technology is probably the most significant development in software technology in 40 years. Like all major baseline technology shifts, this one won't be felt immediately but over a 10-year period. A massive amount of development has to be done, but this is so far-reaching in its effects that it will impact literally everything that's being done with software. You now have the ability to standardize, and that changes the economics of software development, which is where the big bucks are spent...."

[Merrill Lynch & Co.'s Anthony Pizi]"...Multimedia is the future for training. It allows people to learn at their own pace. This technology is very good for training people in a standardized way. A firm's staff might be spread out among five different offices, but they can all use the same training program. Investment bankers might use multimedia to give a presentation to a client and make changes on the fly. But one of the biggest problems is portability. You need either tremendous amounts of memory or a U-haul to lug all the hardware like CD player, CPU and monitor. Live video is on its way, but any time you incorporate video and sound, huge files are created, which have to be compressed. More "affordable" boards are coming out, so end users don't have to worry about losing any data..."

[Andersen Consulting's Robert Gach]"...Rule-based processing and object-oriented development are technologies that firms on the Street could use to steer clear of large applications, which have millions of lines of code and don't lend themselves to flexible new product development. Object-oriented technology and maximizing code reusability — changing something, copying it or adding a variant—as opposed to starting from scratch, would save the Street a lot of development costs and time. Applications would get to traders faster if programmers could reuse packets of logic that are two to three commands long. Applications with only 30 percent to 40 percent of the code are easier to maintain, document, understand and fix. In addition, by using both these technologies, programmers stay in the end user's world and define systems from a business standpoint..."

Hot technologies for the 1990s, Ann Goodman and Jenna Michaels, ***WALL STREET & TECHNOLOGY****, 2/92*

Methods

...There is a creative tension building in the computer industry between the "way we used to do it" and the "way we should do it." Object technology is causing this tension, as the vision of the way we should do it clashes with current reality. There is only one way to resolve this tension we find ourselves in: either pull the vision of object technology towards reality, or pull reality toward the vision...

Object Request Broker—the end of the beginning, Chris Stone, **OBJECT MAGAZINE**, *3-4/92*

...The changing nature of systems, however, including: a move to distributed development and deployment, the increased use of graphical user interfaces, improvements in language and environment technology, widespread availability of classes and libraries, will speed the adoption of object-oriented programming languages. Large systems of the future will evolve on a project or functional basis rather than as multiyear phased systems. As the world migrates to reusable components and the development of organic systems, small methods will once again dominate. A method like IE, based on the mainframe-centric master enterprise model view of the world, will ultimately collapse under its own weight. When objectified versions appear, view them with caution, and ask if the problems they are solving will be relevant to you by the time you complete your enterprise model.

Methodology: Objectified information engineering—the method time bomb? Adrian Bowles, **OBJECT MAGAZINE**, *3-4/92*

Class Libraries

...In the longer term, it will be up to content specialists to define the basic services that classes should provide in a way that makes them more independent of their implementation context...

...In addition to warranties and certification processes, I believe that class vendors should provide testing facilities with every class they ship. Ideally, these facilities would include a complete test bed for automatically sending a full range of messages to the class and checking the correctness of its responses. Providing developers with these testing tools would allow them to double check the performance of any class. More importantly, it would help them assure that any modifications or subclassing they performed had not violated the basic functionality of the class...

...Ultimately, the solution to compensating class creators will be determined by market forces, which usually have a way of defying the most well-intentioned attempts at prediction...

Easing into objects: Developing the object components industry, David A. Taylor, **OBJECT MAGAZINE**, *3-4/92*

...ACIS emerged in December 1989 as an object-oriented modeler written in the C++ programming language. ACIS is a naked kernel of a solid geometric data modeler that is designed to let software applications easily hook into it. As such, it is easily extendable into virtually any application requiring full 3-D support. "It's like the engine of a car: It's the heart of the system, but you're not going to sit down and drive an engine," says [Spatial Technology's Richard] Sowar. Scott Owens, ACIS product marketing director, says that ACIS-based applications are being developed by the four largest object-oriented database companies: Objectivity..., Object Design..., Ontos..., and Versant... He says that within 18 months, 20 of the top 25 CAD companies will have ACIS-based offerings....

Software supports simultaneous engineering, Jeffrey Zygmont, **MANAGING AUTOMATION**, *2/92*

Languages

...Practical studies based on function point analysis show that a 4GL solution to a typical problem is 50 per cent simpler than 3GL solution, but only 15 per cent simpler than an OOPL solution. This is probably because the OOPL's user-defined types are counted, while those built into the 4GL are not. The size and complexity of a 4GL solution grows incrementally with the system. With an OOPL, you just add components to meet each new challenge. Fourth generation and other specialized languages will stick around, if only for cultural reasons (most of today's programmers cut their teeth on them). The world is unlikely to rally around a single OOPL. Technical and financial developers use C++. MIS shops prefer Smalltalk or are waiting for object-oriented Cobol. Ironically, object-oriented 4GLs now accompany some OO databases to help them integrate objects from multiple OOPLs.

Software & Systems: Fourth generation heyday at an end, Craig Hubley, **COMPUTING CANADA**, *2/17/92*

...At press time [Sequent's Larry] Evans didn't want to elaborate on Ptx/Object, but according to documents obtained by UNIX World, the product will combine a parallel version of the ParcPlace Objectworks/Smalltalk development environment and a parallel implementation of Versant Object Technology Corp.'s object database management system. This will enable users to tap information from legacy systems and convert it to objects...Because the object-oriented market is in its infancy, Ptx/Object won't generate any short-term income. As the object- oriented market becomes more mature, however, Sequent's early entrance will help it, say analysts...

Sequent's software cure, Gary Andrew Poole, **UNIX WORLD**, *3/92*

tion path from our investment in methodologies and information technology, they will inhibit the development of the market for their own products. At issue is the existing financial investment, business risk of a wholesale conversion to a new technology that we do not as yet know how to manage, and credibility problems that arise when IT managers tell their senior management they need to change from something they have just spent millions of pounds implementing, such as IE.

IE has brought to many of us the challenge of enterprise modeling. Enterprise modeling is of interest because a model of the business makes it easier for people to understand the business. Object modeling, with its more natural way of reflecting the business of the enterprise, could provide an important discipline for thinking about the business, regardless of whether or not a system was actually being built. The very availability of an object model could make business reengineering more tractable. One could think of business objectives being translated into specific states that objects modeling the business must attain. Business objectives can finally translate into "something" that can be manipulated on a computer—long an issue with IE. This is a vision, and certainly not a trivial matter, and is complicated by accommodating strategic planning as well as shorter-term objective setting. We certainly need a better understanding of how business objectives and strategy setting would be best done in conjunction with O-O enterprise modeling.

There is evidence to suggest that the identification of stable domain objects is better carried out by a group process. In more detail, a small team is established, is armed with a sound description of the business domain, and a series of group sessions follow managed by a good facilitator. To support this process, there is a need for guidelines on:

- What should be the team make up and role definitions? In particular, what is required of the facilitator and modeller roles?
- What is an effective format for the modeling domain description and what is the process for establishing it?
- What are the recommended processes and techniques to be used by the team?

Getting the right level of abstraction is fundamental to the quality and stability of enterprise models. Object orientation does seem to enable the supply of high-level, flexible frameworks for particular industries that are well thought out, meet these criteria, and allow for customisation to specific needs. This promise, itself, raises a lot of new issues for the IT division.

OBJECT CASE & METHOD

We want a competitive market for users to mix and match object-oriented analysis and design methods and tools within a framework for the O-O paradigm that enables production of extensible systems and maximises reuse of previous work at the highest practical level of abstraction (e.g., design and enterprise architectures).

Object CASE products should enable project manageability, i.e., communicability and predictability, with appropriate control. These products should exploit the consistency of the O-O paradigm to enable intuitive use by business modellers. Object CASE needs to support methods that allow analysis and design, from identification of business objectives through to a clear end point that is defined as complete when all objectives for that business subject are provided for by the modeller.

> ❝ ***We will need much more sophisticated support for layering models than has been available from past generations of CASE.*** ❞

The CASE products will have to provide for iterative development of dynamic application models covering business objectives (requirements/object communication models), concepts (domain model) and events/processes (behaviour or dynamic model). We will need much more sophisticated support for layering models than has been available from past generations of CASE. Further, abstractions are needed to capture business definitions and domain concepts, with emphasis on semantics and the exploitation of encapsulation (information hiding) and class inheritance (specialisation/generalisation). Such support would optimise reuse or modification at the semantic and diagram level. Support will be required for ensuring the coherence, completeness and consistency of the models and for the associated documentation. The products would be supported by an extensible repository organised to support reuse and suitable for browsing.

From what we can see today, the model of events and processes shall encompass object life histories by states and be arranged to assist exhaustive verification. Correctness of process behaviour would be checked by pre- and postconditions, preferably in semantic form. Business objectives would be treated as requirements to achieve a given state and represented by an object com-

munications model. Also, business objectives would be semantically traceable to process level and available as use cases for testing.

In short, our requirements are for a detailed, CASE-supported method enabling early prototyping and encouraging the reuse of code and design. It will have to cover how to set up and run an O-O development project, what processes we will execute to maximise the scope of reuse, an appropriate lifecycle with deliverables, and quality considerations and processes.

DEVELOPMENT ENVIRONMENT

Given our emphasis on reuse, a development environment will be needed that is rich in class libraries of reusable code, and provides tools to support fast prototyping and intelligent browsing. If people cannot browse easily, reuse will be harder than reinvention and reuse simply will not occur. Further important features of a good O-O development environment, regardless of the actual programming language involved, are:

- incremental compiler/linking
- change-management support
- single and multiuser support
- runtime inspectors
- message-passing protocols
- storage management
- dynamic binding
- type-checking facilities
- performance analysis aids.

> ***Objects are not applications. They are resources. Building resources requires a fundamentally different organisation than we have today.***

Further issues arise in relation to distributed development environments:

- What are the problems involved in a development environment having a distributed architecture?
- What are the issues arising from encapsulation, class hierarchies, and reactive inheritance associated with distribution?
- How do you manage the reuse of resources in a distributed environment?
- How do you manage C++ classes across a client/server architecture?
- What is the current feasibility of moving objects across a distributed system regardless of platform or language?

The capability of migrating across different hardware platforms is increasingly a requirement. Many large companies have a considerable mix of languages and code that are an impediment to integration and reuse. Choosing a language for an emerging technology threatens to compound this problem, particularly if it does not emerge the winner. Furthermore, many companies have increased developer productivity with a high-level language (e.g., Telon). Falling back to the complexity of C/C++ is not seen as attractive by large IT divisions, despite the apparent popularity of C++ in the current marketplace.

REUSE OF COMPONENTS AND DESIGN

Most large IT organisations are broken into separate project teams. This produces cottage industry cultures, with each team focused only on its own project even where optimisation of an individual project works to the detriment of the company as a whole. Project managers are paid to deliver their projects. They are not rewarded for slowing down or complicating their own projects by involving other projects—even though such involvement is required to deliver the more effective corporate data architectures that are the goal of many IT divisions. The incentives do not drive cross-project resource commitments.

For reuse to work, we have to change this culture from one of building applications to building basic business resources. Objects are not applications. They are resources. Building resources requires a fundamentally different organisation than we have today. Corporate data architecture has not really worked because of IT culture and organisational structure. O-O provides a vision of how this could be brought about because it provides a basis for an organisation that delivers basic resources instead of separate applications.

Clearly, changing the IT organisation is fundamental. Current staff reward systems are delivery based and mitigate against reuse. We need to determine how the staff assessment/reward system should be changed to support reuse, and how the IT organisation should be structured and managed. Leadership, group, and supporting roles will change and new roles (such as object librarian) will become essential. As this is a cultural change problem, what we need is a model for change management. This is not a problem unique to O-O and is at the heart of the limited success achieved with past change efforts.

F.Y.I.

Excerpts from leading industry publications on aspects of object technology

In the Industry

...The constant refrain heard among the audience [at PC Forum] was this: We shouldn't talk about objects because nobody agrees what they are, users don't see them, and nobody really has them. The notion was that objects are sort of the industry's dirty laundry, so they really shouldn't be hung out in decent company...We disagree. Objects should not be swept under the rug. We know that the notions of object orientation have been around the industry since the early 1970s. But we believe that the technology infrastructure is only now becoming sufficiently powerful and sophisticated so those notions can be implemented in everyday systems. And we think that development will change everything about computing for our readers. And we mean everything, including what skills you need to be successful, what kind of products you should invest in, the methods you use to evaluate those products, the expectations you have (and set for your management) about how fast you can implement new applications, and the approaches you choose to adopt in designing your systems...A key benefit of object-based systems, for instance, should be to move the locus of responsibility for applying technology from the vendor...to the customer...

Editorial: Object-oriented technology needs to be thrashed out, Stewart Alsop, ***INFOWORLD****, 3/2/92*

...Today's pet technology, object-oriented programming, certainly will speed development and make it easier to maintain programs. But creating applications that integrate voice and pen input and enable collaboration among groups will require new paradigms no matter how much object-oriented programming is thrown at the problem. Ultimately, for these new technologies to take root and succeed in transforming the computing landscape, we believe that the hardware makers, operating systems companies and application developers need to collaborate with the users. Without input from the real world of users, there will be far more misses than hits. Placing users at the center of the design and development process will be the shortest route to success.

Editorial: Today's users key to tomorrow's technology, Henry Norr, ***MACWEEK****, 3/9/92*

...(Jim) Manzi (CEO - Lotus Development Corporation): Object-oriented programming is exceptionally important. From a technology standpoint, it means better software-making. It will also turn into better products on the other end for the users...It will be qualitatively better in many ways because of certain design things that let you directly touch and move stuff and indirectly manipulate things on the screen. It's just a better user experience...It's also in its relative infancy. UPSIDE: What has Lotus done in the brave new world of objects? MANZI: Improv as a product and NeXTStep as an operating system are the most object-oriented stuff in the world today. cc:Mail is completely re-architected in C++. It is probably the most widely used object- oriented product in the world today. We have yet to find the need to put a banner across the top left-hand corner of the box that says, Now Improved with Objects. UPSIDE: Perhaps you should. MANZI: But the user doesn't care. The user shouldn't care. Borland has done a great job of pretending they invented this stuff. It's been around for a couple of years, and some people have actually done stuff in those couple of years before Borland. But, life goes on....

What is it about Jim Manzi?, Rich Karlgaard, ***UPSIDE****, 2/92*

As IBM Corp. and Apple Computer Inc. last week named a management team for their object-oriented venture Taligent, archrival Microsoft Corp. outlined plans to develop an advanced object-oriented environment around Windows. Microsoft chairman Bill Gates provided a long-term view of the company's system software strategy, which entails increasing the functionality of Windows and making it a linchpin for enterprise computing. Ultimately, the goal is to create an object-oriented software environment that will sit on top of Windows operating systems such as NT and DOS...The object-oriented software environment, called Cairo, is expected to come to fruition in the same tame frame as Taligent's development efforts, about mid-decade...

Microsoft reveals object plan, Amy Cortese, ***Systems & Network Integration****, 3/2/92*

The Open Software Foundation's Distributed Management Environment efforts could be delayed by up to two years due to infighting between OSF and some of its technology providers, an IBM network management official said last week...Jeff Thiemann, HP's OpenView business development manager, said the "Clash of ideas" stem from the DME's mandate to tie together technologies that support current industry-standard management technologies or products, such as SNMP standard and HP's OpenView, with promising new and developing technologies such as Tivoli's object-oriented programming approach. Thiemann said that those involved in the DME face two alternatives. The OSF could stick with its ambitious original plan, which called for the release this year of an integrated DME platform that includes specifications for the Consolidated Management API—developed to address current management technologies—as well as specifications for object-oriented systems administration under DME. Or, Thiemann said, OSF could stagger independent releases of the CM-API, DME's object-oriented technology, and, even later, the integrated DME implementation....

Infighting may delay OSF's DME, Laurel Nelson-Rowe, ***Open Systems Today****, 3/2/92*

CPSR: A BRIEF HISTORY

Computer Professionals for Social Responsibility was formed by Severo Orenstein and seven other researchers at the Xerox PARC research facility in 1981. Incorporated in 1983, the group was a noted critic of the Strategic Defense Initiative. Many prominent computer science researchers joined the organization, agreeing that many of the long-range goals enunciated by then President Reagan were either dangerous or of dubious functionality and would increase the danger of accidental nuclear war. CPSR obtained funding for investigations of autonomous weapons development. SDI-related work reached a peak in 1988, following the publication by Harcourt, Brace, Jovanovich of Computers in Battle: Will They Work? edited by D. Bellin and G. Chapman.

By 1988, local working groups, composed mostly of programmers, industry researchers, and university professors, were active around the US. These working groups gathered bibliographies and conducted collective research leading towards the publication of papers and notes on workplace monitoring, civil liberties, privacy, support for nonprofit groups, and other issues. Members donated their time to develop free software to meet the needs of other community and peace groups. Congressional committees began inviting CPSR to provide expert testimony, the most notable example being the invitation of Congressman Don Edwards for testimony on the National Crime Information Computer II proposed by the FBI. CPSR testimony played a major role in the revision and restriction of the original proposal.

CPSR has grown to include approximately 2,500 members and some 20 chapters. Similar organizations exist in Great Britain (Computing & Social Responsibility, Gordon Blair, University of Lancaster CS Dept., Bailrigg, Lancaster LA1 4YN), West Germany (FIFF, per addressee Helga Genrich, Im Spicher Garten #3, 5330 Koenigswinter 21, FRG), Australia (ASRC, Graham Wrightson, Dept. of Computer Science, Newcastle University, Newcastle, NSW 2308), and Canada (IMPUT, Box 248, Station B, Ottawa, Ontario K1P 6C4). CPSR received the first grant awarded by the Electronic Frontier Foundation: $250,000 to fund the civil liberties program from the CPSR office in Washington, DC. The Rockefeller Foundation recently awarded a major grant to the CPSR "21st Century Project," which seeks to demonstrate the utility of advanced technologies for solving the worlds most pressing problems of ecology and sustainable development. Dr. Eric Roberts, professor at Stanford University, is the current CPSR president, succeeding Dr. Terry Winograd, another Stanford professor and prominent AI researcher.

CPSR may be contacted at: PO Box 717, Palo Alto, CA 94302, 415.322.3778, or at 666 Pennsylvania Avenue SE, Ste. 303, Washington, DC 20003, 202.544.9240.

ward the use of electronic media, our expectations of communications privacy are not being enforced nor are the responsibilities of vendors clearly spelled out.

Why are there so few and such weak laws regarding privacy and security on the electronic frontier? And where are our elected representatives during all this? How many of them even understand what email is, anyway? Although the *Computers, Freedom, and Privacy* conference was held in our nation's capitol, elected representatives were absent. It is up to you readers to help wake them up. This points to the need for public discussion and involvement by the computer literati. These issues will become more and more important to us as the electronic age develops and the object-based electronic office becomes a reality. Interconnected computer networks, constantly expanding client–server systems, really do consist of object instantiations floating through a new frontier. We'd better be sure they're doing the job we intended, or we, the unaware, will pay the price in the gradual erosion of the freedoms we have won over the last 200 years. ≡≡

Reference

1. *MacWeek* 6(13):3, 1992.

Mark your calendar

OOP

Plan ahead for the largest object-oriented conference and exhibition held in the U.S. in 1992.

Object EXPO

THE NATIONAL CONFERENCE AND EXHIBITION

JUNE 1–5, 1992

THE SHERATON NEW YORK

For a detailed brochure call 212•274•9135 or Fax: 212•274•0646

An organisation structure that focuses on resource-building projects must be supported by a reuse-oriented development method and tools. Among the pertinent issues that arise are:

- The need for a set of evaluation techniques/guidelines to determine when it is appropriate to create or use a library.
- How should the use of libraries be licensed and billed?
- What is the range of object types that are candidates for the library? Candidates fall in some very diverse categories: generic application objects (e.g., payroll), domain objects (business model), and component objects (GUIs, etc.).
- How should these be managed in their different forms: design specifications, source code, object code?
- How do we assess the availability and suitability of external libraries?
- What are the risks of using purchased library code, and how are issues of ownership, version control, and security to be managed?
- The internal construction of good libraries requires up-front investment. How should libraries be financed and cost apportioned given that they are enterprise assets whose development cost may substantially exceed the needs of any particular application development project?
- How will O-O libraries coexist with existing applications, data dictionaries, and conventional code libraries?
- Which standards are essential to enable reuse, e.g., object model, message protocols, etc.? When and how are they likely to emerge?

Reuse poses both cultural and technical challenges, and is not simply achieved as an instant byproduct of O-O. Some evidence suggests that designing for reuse is as difficult to achieve as the shift in thinking required to understand and use O-O in the first place.

DATABASES & OBJECTBASES

For commercial organisations, there are no obvious immediate applications for object-oriented database management system (ODBMS) technology, or, more accurately, *objectbase* technology—particularly as the move to relational database management systems (RDBMSs) is far from complete—and there may never be. There are probably specific niches in specific organisations for which objectbases will be immediately useful. However, organisations will continue to select products based on both their suitability for the task at hand and fit with existing technical architecture. Objectbases are currently too immature and do not fit well enough with existing systems—particularly as there is a lack of mainframe-based products—to replace enterprise-critical data systems.

Remember that OIG members depend on and maintain enterprise-critical systems that are typically 20 years old, have over 600 man years of development effort invested in them, support upwards of 40,000 online terminals worldwide, and are in many cases written in assembler because that was the available, proven commercial technology when the systems were developed. This is the context into which we are trying to introduce O-O. The challenge is finding areas where the objectbase is useful.

The encapsulation of process and data and their storage in a single mechanism makes the objectbase an applications engine, not just a data store. This is an evolutionary step towards intelligent databases, as the inclusion of knowledge-based rules within the objectbase makes the it an active engine rather than a passive store.

This may make the future for objectbases bright, although, from the perspective of the OIG, this technology is simply not ready to provide the level of reliability, support the number of users, deliver the security control, or handle the volume of activity our businesses experience. There is, in our view, a substantial difference between today's objectbase and a truly commercial product suitable for the kinds of business problems we must solve in the normal course of daily business operations.

Our requirements for industrial strength objectbase products include:

- A vendor-independent SQL analogue such that portability across objectbase products can be achieved by adherent applications in the same manner as is available with RDBMS products.
- Rich schema support online including the ability to change attributes, methods, supertypes, and subtypes, etc. on a running objectbase. An objectbase is of little value if it cannot be changed without a complete, out-of-service unload, rebuild, and reload. The companies involved in the OIG regularly deal with databases sized in the tens of millions of records; objectbases would be similarly sized.
- Change notification to applications so they could be self-protective.
- Access from conventional languages like PL1 and COBOL.
- Access to data stored in DB2 and IMS databases, given that much enterprise-critical data may never be moved into the objectbase, but would be needed to populate characteristics of objects stored in the objectbase.
- High-speed transaction processing of the level needed for airline reservation and banking systems—thou-

sands of nodes, tens of thousands of workstations, and millions of transactions.

- Performance-monitoring tools for isolating and solving the kinds of problems that inevitably arise in tuning complex systems.
- An underlying theory of ODBMSs to encourage more conformity in vendors' products.

> ***At issue is the feasibility of introducing O-O and distribution in parallel.***

An industrial-strength objectbase would feature the kind of performance and support we have today in products like mainframe-based PARS high-speed file management and IMS. Even relational database management systems like DB2 are not providing the following capabilities to the level required by, say, an airline reservation system:

- integrity
- security
- back up online
- version control
- multiuser access
- multiple OOPLs support
- conventional programming language support.

As OIG members are significantly concerned with migration path issues, we need to understand to what extent a relational database can be extended with O-O functionality, yet still deliver the benefits O-O promises. This appears to offer an easier migration path, but is it worth it? What are the limitations of extending relational systems? How effective will this be given that relational has seen only limited adoption in core systems today even though it has been available commercially much longer than O-O?

O-O gives us the ability to create objects with less regard for exactly where the attributes are stored. Instead of IT focusing on applications that manage their own data, or building databases that are inherently weakened by separating the processes and the data, O-O allows us to build business components that match workplace needs and store the actual attribute values in our existing legacy systems if need be. This, however, presumes interoperability between object management systems and today's DBMSs.

Large companies need to preserve their existing investments, and a migration path (e.g., access to RDBMS, IMS, etc.) is essential for the widespread adoption of the objectbase by large-scale IT users.

DISTRIBUTION

A typical bank in England will have 3 data centers, 100 regional centers, 3,000 branches, and 25,000 workstations operating across both wide- and local-area networks. What such an industry needs for distribution is simply not available from O-O products as yet. For O-O to be adopted by the financial community, and especially by the major clearing banks, it must be possible for objects, or at least their methods and inheritance mechanisms, to be replicated at 3,000 or more branch end points.

At issue is the feasibility of introducing O-O and distribution in parallel. Distributed systems are complex using today's relatively well-understood technology, but how suitable are current O-O methods for such tasks? Can O-O be exploited to overcome current limitations and, if so, what are appropriate guidelines? We need to understand in more detail:

- How suitable current O-O methods are for distributed processing now. Once we understand how to handle simple mainframe-to-PC distributed processes, we can undertake full client/server, LAN mainframe connectivity and, eventually, peer-to-peer.
- If introducing O-O and distribution in parallel is either feasible or desirable and what guidelines and migration strategy are needed on this parallel introduction to ensure complexity does not get out of hand.
- How O-O methods can be incorporated into vendors' preferred approaches, such as IBM's SAA CPI-C, and which components of current methods are appropriate or inappropriate for O-O development given that current development methods tend to reinforce the present application and team structure that, as we have already noted, inhibits effective reuse.
- How current program-to-program communication protocols and remote procedure calls can best be used to relay messages between objects in a distributed environment, and what the impact of OMG's Object Request Broker will be.
- What the impact of ISO's Open Distributed Processing standards proposals will be in relation to O-O and how they might conflict with, overlap, or augment the OMG standards.

Our further requirements are essentially those for

have to allow provisional changes to documents and data through the interaction of professionals who cannot be expected to be present in either a particular physical location or at a specific time. Distributed databases will have to interact not only over a local network, but over the entire organization and across national boundaries. Client–server applications that we already see beginning will have to be extended with a broader view. My prediction is that we will see the client–server concept extended so the server is no longer thought of as simply "the mainframe." The server will be whatever hardware/software combination is the primary processing platform for any particular application. From this perspective, it is clear that the future of client–server frameworks will be successful only when fully integrated with object-oriented technology concepts that will allow the client to send and receive messages from any server in the corporate network.

In the coming object-oriented, client–server networks in the electronic office, it will be crucial to guarantee both the privacy and authenticity of messages. As network structures and operations become invisible to developers as well as users, standards for privacy and security become more important. Because developers will be less likely to look at the inner workings of purchased and reused class libraries, how will their managers know the implications of what was purchased? How do we guard against weak security schemes or even worse, hidden Trojan horses? Does purchasing off-the-shelf class libraries without source code increase the risk of privacy and security violations? Recent incidents of virus contamination in commercial, shrink-wrapped, PC software packages make this a very real problem.

Privacy is easier to address today than the issue of security and safety. Privacy can be guaranteed through strong encrypting schemes, the most promising of which is presently a public key scheme based upon large factorial numbers as the keys, such as in the RSA scheme. Researchers at NeXT, Inc. and universities are working on a new scheme based upon "elliptical curves," which also holds promise. David Chaum and Digicash, Inc. have proposed authentication methods that preserve privacy, and public key schemes can provide authentication for more general cases where anonymity is not required. We are finally seeing some of these encrypting schemes in commercial products—you will shortly be hearing about some in new networking announcements from Microsoft and Novell, for example.

But for some reason, our government lags behind. The National Security Agency is dragging its feet, causing delays in the announcement of new encrypting standards at the National Institute for Standards and Technology. It was also revealed that the NSA has been engaged in secret discussions with the Software Publishers Association, with a view towards persuading the latter to drop their support of proposed legislation that would permit the export of encrypting software without NSA approval. This was confirmed in many industry periodicals (e.g., MacWeek[1]).

The government attitude today extends beyond gentle persuasion, suggesting that their objection may be self-serving. In just one example, all the computer equipment and the electronic bulletin board for a private business was seized by the US Secret Service. The offender was Steve Jackson Games, Inc., producers of various gaming systems. The alleged offense was information left on their computer by the author of a new fantasy novel on cyberpunks, which the Secret Service holds is a manual for electronic terrorism. No charges were filed, but the Jackson business almost went bankrupt and has laid off half its 15 employees.

Why do "authorities" think they have a right to go through electronic files when they couldn't do the same

> ***The government attitude today extends beyond gentle persuasion, suggesting that their objection may be self-serving.***

to paper ones? Government agencies are not the sole offenders. I was recently told by one bitnet user how, while sending a message across the country, the system administrator broke in with a comment which made it obvious that the email was being read by him. And the verbiage used by the administrator was at least offensive, if not obscene. Recently the employee of a major computer manufacturer was reportedly dismissed as a result of intercepted personal electronic mail. As computers move towards the network-messaging model implied by object-oriented technologies, we need some way to censure such actions and prevent their recurrence.

No major network, and no major bulletin board service, from Bitnet to CompuServe to Prodigy, makes any effort to ensure the encrypting and privacy of email. The major commercial vendors do make a promise that no employee shall routinely inspect personal email, although we are never told how that will be enforced. This leaves us in a precarious situation. As we increasingly move to-

PUBLIC POLICY

Objects, security, and privacy on the electronic frontier

by David Bellin

For many of us, object orientation is part and parcel of the electronic network that is increasingly tying together people and data scattered all over the world.

The net is a complex, growing artifice spawned from non-object code. However, the net has always had nascent within it an object-oriented view of the world. Consider, for example, the matter of network addresses. The non-object view would force the specification of an intricate physical gate path through the net to the recipient. In contrast, the much more successful encapsulated view merely states the user's name and colloquial location. The net takes care of the methods and data for physical delivery.

Unfortunately, our technical view of this networked world often overlooks crucial considerations of privacy and freedom of expression in the electronic frontier. This has led to debates over who can say what on the nets, concerns over the privacy of electronic mail, and legitimate fears over the security of personal data and business transactions.

These were among the issues addressed last March at a seminal event that is sure to be the forerunner of many discussions on such topics. *The Second Conference on Computers, Freedom and Privacy* was held March 18–20, 1991, in Washington, DC. Sponsored by Computer Professionals for Social Responsibility and the Electronic Frontier Foundation, the meeting was cosponsored by a long list of professional societies and companies such as AutoDesk, Apple, and the Association for Computing Machinery. It brought together the most influential players in the computer world, along with their counterparts in industry and government.

What emerged was a story of a network of computer connections that has been expanding and connecting faster than our legal system can adapt. There has been a concomitant growth in computer bulletin board services, into which anyone with a modem can call. Debate at the conference pointed out many legal gray areas that need action. For example, is computer electronic mail afforded the same privacy protection as printed mail via the postal service? Do these protections hold only when email is transferred between networks, or should they also hold within individual private networks and bulletin boards? Why don't network vendors make easy-to-use encrypting techniques available for private email? Who should regulate this industry, if anyone?

Many conference participants wanted to see government get out of the business of imposing encrypting standards. I, in contrast, call on government to stay active—but to promote policies quite the opposite of those currently in force and instead take a leading role in the promotion and dissemination of cryptographic techniques in areas beyond electronic funds transfer (wherein they are currently involved), specifically in electronic communications privacy.

One of the best encrypting schemes proposed to date is the public key system offered by RSA Data Security, Inc. of California. Also worthy of consideration are the important technical ideas promoted by David Chaum, a prominent computer scientist. Chaum has developed a foolproof scheme to ensure that the privacy of your credit transactions is guaranteed while security is not compromised. The wonder is that financial institutions seem so resistant to the adoption of Chaum's ideas. To ensure security in the electronic world, our government should require network providers to have encrypting facilities available to all users of electronic mail. At the conference, this generated strong audience approval indicating the existence of a demand for such protection.

The electronic office will have to be based on objects if it is to succeed: Cooperative work arrangements will

David Bellin is a noted author and consultant. His fifth book, Software Maintenance: The Small Systems Management Guide *has just been published by Prentice Hall. Dr. Bellin teaches computer science and Smalltalk at Pratt Institute, New York. He welcomes reader comment and can be reached at (718) 499-6443.*

open distributed processing. An object distribution system that has industrial strength would have multivendor support and standardisation across platforms. It would provide an operational execution environment and all necessary management and control facilities. It would be transparent regarding location, access, concurrency, replication, failure, and migration. Management and control facilities would be provided for sending new objects to operational nodes. Version control, operational triggers, object data partitioning, partial object distribution, and class hierarchy distribution would be essential for large-scale use.

The trend in IT is towards open, distributed processing allowing the business to place its processes and data according to the primary business needs. Object-oriented systems to date have been developed as single user systems, but if the benefits of O-O are to be made available to the user community as a whole then distributed, multiuser systems employing objectbases at their heart are essential. This is not to say that benefit cannot be gained from O-O systems today. Clearly O-O can support reach and range applications, modeling the workplace, integrating systems, etc. There is simply a sizable gap—particularly in the area of distribution—between what we can do today with O-O and what would be needed for O-O to replace existing mainframe technology.

MANAGEMENT

The fundamental changes in the software production process raise many issues for management for which new methods and techniques are needed. As we have already said, moving from today's environment to take full advantage of object orientation will require changes in IT culture. Whereas today's environment can be summarised by:

- system-based solutions that focus on meeting a specific requirement
- most code is built from scratch
- reusing other people's code is difficult, risky and ad hoc
- if the requirements are correctly specified, the system will do its job
- the first major delivery is the hard one; after that, we can usually cope with change—at a cost and given enough time
- infrastructure activities, such as conforming to corporate architectures, slow projects down in terms of meeting clients' needs
- code is the real asset

the object-oriented environment will be characterised by:

- IT solutions based on an inventory of reusable software assets
- most code drawn from the inventory
- inability to specify requirements recognised as a fact of life: an 80% fit will be adequate because the world changes too rapidly to achieve 100% fit profitably
- management of the inventory and its contents becoming a vital corporate function
- provision of high-quality reusable software for the inventory becoming a vital specialist role in IT
- design being the real asset.

Object orientation brings the Industrial Revolution to the software industry. Potentially, applications of the future will be built from reusable components that are assembled to meet particular needs. These reusable components, or objects, will be assets of the enterprise just like product designs in a manufacturing industry.

Object orientation moves the emphasis from the computer to the user, from the demands of the hardware and systems software to the real world in which the business operates. The computer model of the business, no longer separated into function and data, now matches the real world with much greater fidelity. Recognition that change is the norm replaces the old idea of systems being "finished." The fact that the model corresponds closely to the real world means that changes to the system should be on the same scale as change in the real world.

In general, we need more detailed understanding of the impact of O-O on skill requirements, training, and management responsibilities.

Managers will need a different perspective to most effectively utilise O-O—one more akin to resource (asset) management. Managers will be responsible for a more diverse asset list: people with specific human skills (e.g., experience levels, training requirements, recruitment, and consulting), hardware (PCs, UNIX workstations, LANS, and WANS), and software (code and design asset base, development tools).

Managers will also have to change the way they view software, making software components a value instead of a cost by making them reusable. When used once, software components may deliver value, but cost recovery is more difficult because the entire development cost must be amortised on the first use. Reuse changes this equation entirely. We have no experience in managing software at the design level as an asset.

O-O also changes project management, budgeting, and resourcing in other ways. Managers must come to terms with up-front investments by their projects in resources with long-term value instead of simply budgeting to build components used in isolation. Investment in assets will not be optional. Only part of this is a tech-

nology problem. The more serious problem is cultural and managerial.

Although organisation changes have been addressed elsewhere in this article, a summary of the areas of organisation change required to fully utilise O-O makes clearer the impact of this new paradigm:

- infrastructure (corporate data architecture, quality, training, technical support, etc.)
- management structure
- team structure, sizes, and roles—new roles may be central or distributed (domain analyst, class librarian, etc.)
- interteam communication
- hardware homogeneity/heterogeneity and interconnection strategies
- client roles

> ***We need to be able to show that the inventory of business objects being developed is in fact worthwhile.***

As well, specific procedures will have to change, raising questions about:

- team selection/building
- management strategy, predictive vs. experimental (risk based)
- costing, planning, monitoring, control mechanisms, risk-management
- configuration management and change control
- incentive and reward schemes to encourage reuse
- system roll-out and evolution
- metrics and project progress reporting

Managers will be expected to implement migration strategies. We have more questions than answers at this point, as many Group members have only recently implemented some of the "previous generation" approaches like IE and technologies like RDBMSs. While answers to these questions are less problematic on very small scales, in large IT divisions as ours the situation is somewhat more challenging:

- How do large organisations introduce O-O—how much, where, and when?
- How do we manage the interface(s) between concurrent O-O and conventional development projects?
- How do we reengineer or phase-out existing stocks of system code?
- How do we manage the O-O learning curve (variously estimated between 6–18 months)? Should we start with OOA/OOD training and then teach a language or is there any wisdom to starting with a language migration as a teaching vehicle?

Ultimately, we require an object-oriented management model that can support this future vision and addresses the commensurate change in business objectives, resource management, organisation, procedures, and migration strategies.

METRICS

Our production systems today are process centric. Progress is measured by the point reached in the waterfall lifecycle. Object orientation will result in production systems that are product centric, in which an important measure of progress is the accumulation of standard, interchangeable, reusable components. We require useful metrics and systems that enable their collection, collation, and presentation.

Suggested metrics for collation are:

- Project size, as measured by numbers of classes, instance objects, methods per object, lines of code per method, users, and terminals.
- Development time scales such as the number of man days spent on modeling for analysis, modeling for design, coding, testing, and user training.
- Productivity, in change over time for analysts, designers, programmers.
- Reuse, as in the percentage of code reused, customized, and coded from scratch.
- Performance, as measured by batch transactions in 24-hour period, online transactions in 24-hour period, or response times for online transactions.
- Quality, reliability, extensibility

Sound metrics are essential to measure project progress (especially on large projects) and verify return on the investment in O-O IT. We need to be able to show that the inventory of business objects being developed is in fact worthwhile. The business object inventory raises many of the same issues as traditional hard-goods inventory management-cost control, cata-

dispensable. The reason MIS departments do not see the need for cross-functional, dynamic systems is that their vision is obscured by years of building applications where the emphasis was on line data collection, followed by stashing gigabytes of records into files and churning them in batch overnight.

HOTLINE: If I understand, just getting started is a real problem. MIS needs a toe-hold—some familiar ground on which to take the first step. How can MIS find good first applications?

KROL: The first object applications should be models of existing systems. These models can be used as living documentation—much more vivid and useful than dusty paper binders. Models are also a good way of getting in touch with the fundamentals and understanding tangled layers. For example, a claim-processing model in an insurance company can vividly illustrate the slow progression of a claim in a bureaucratic labyrinth of an insurance company. Models can help to identify the bottlenecks and opportunities for streamlining the whole process. They can also display multiple variations of downsizing and rightsizing scenarios. A financial services company engaged in the process of reengineering is using Digitalk's Smalltalk/V to breath life into its futuristic business models illustrating new ways of conducting business a few years from now. At this point, the majority of these models do not have direct relation to application development. Their audience is high-level management, and their value is almost exclusively inspirational. In time, the new generation of smart CASE tools will be able to translate those models into real applications. The demand for dynamic business modeling/simulation tools will be fueled by end users who have a hard time relating their business objects to the boxes and arrows of entity-relationship diagrams.

HOTLINE: By using objects to understand our own MIS systems, will we learn how to better understand our customers' business?

KROL: Absolutely. By modeling MIS systems, professional programmers are liable to "get ideas" about business. The interest in using object-oriented tools for business modeling is growing. Recently, Gemini Consulting teamed up with ParcPlace Systems to do joint development in Objectworks. The reason Gemini chose object-oriented software is that it was looking for ways to accelerate the process of change. Gemini became convinced that using ParcPlace's Objectworks would eliminate the lag-time between the alignment of a client's business and information strategies, dramatically improving speed and quality of major information system revitalization initiatives.

HOTLINE: Is MIS in the way of business reengineering and the adoption of object technology?

KROL: The more interesting question is whether MIS will actively participate in business process reengineering. Will it have the courage to perform the archaeology, discovering fundamental truths about business, as opposed to continuing with the old familiar things it has been doing for almost a quarter of a century? Some companies are serious about this.

HOTLINE: Do objects empower the users to reengineer the business by themselves where MIS fails to change? Can the users effectively use object technology in spite of MIS, or does this still have to be a team process?

KROL: Users cannot go it alone at this point. Object technology provides insights for both developers and end users, but requires a great deal of technical expertise. Do not confuse object technology with the 4th-generation languages. For the users, objects are an easier way of mapping real-life stuff into electronics. For MIS, objects mean more technical kinds of programming with an emphasis on business.

HOTLINE: Is there a "typical" object application in a commercial world?

KROL: The majority of object-oriented commercial implementations have been along the lines of advanced information synthesis. Such applications bring coherence and have a twofold purpose: they provide transparent access to heterogeneous resources and recreate relationships among such business objects as customers, sales, and products that have been lost in disparate applications built over time. In the past, these types of abilities have been closely associated with Executive Information Systems, and were considered an executive prerogative. In the future, driven by organizational perestroikas, collapsing bureaucracies, and work group management, companies will broaden the scope of such applications from easy-to-use, pampered executive environments to a universal, intelligent front end to corporate environments.

HOTLINE: What do you see as a major contribution of objects to the upcoming decade?

KROL: The major contribution is the ability to bridge the gap that still separates applications from systems, business professionals from MIS, and men from machines. Objects hold a wonderful promise of enabling us to build applications architected for the human race. This is the ultimate goal of business reengineering. ≡≡

row windows of opportunity, short product life and development cycles. Objects will enable the throw-away systems that Peter Drucker has been talking about lately. Another example is a pipeline company that recently built a Smalltalk-based post-production prototype of their ill-performing, multimillion dollar expert system—only to discover that the reason for its poor performance was a number of unnecessary layers that replicated the old processes. In this case, the exercise of building a Smalltalk hierarchy helped this company to identify and eventually streamline outdated operations that had been part of their business for many years.

HOTLINE: It seems to me that you're saying that deep thinking about our businesses is long overdue, but you're calling it object analysis instead of business analysis.

> ***CASE and objects is a marriage made in heaven. Unfortunately, it has not been consummated—yet.***

How does this object analysis differ from really good business analysis?

KROL: Maybe it's not that different. A good analyst *can* get to the bottom of the problem. However, even if the analyst does discover those business fundamentals programmers cannot implement this information with traditional technology because their tools lack the necessary framework to hold the bigger knowledge.

HOTLINE: For MIS, does this mean moving away from the CASE tools they have been struggling to adopt and taking on both new tools and a different role in the business?

KROL: CASE is not so well entrenched in American companies that they would have to wean themselves from it. Meta Group estimates that only 10% of corporate America really uses CASE the way it had been intended, from analysis to code. The majority of companies are using point solutions. There is a certain juxtaposition between business reengineering principles—looking at business from a cross-functional perspective, eliminating assembly-line approach—and what "big" integrated CASE came to represent. CASE is at least partly responsible for institutionalizing old and heavily structured application development processes that have plagued us for years. Hammer has compared CASE to rearranging the chairs on board the sinking Titanic—even with all the chairs in neat rows, the ship was going down anyway. CASE and objects is a marriage made in heaven. Unfortunately, it has not been consummated—yet. Object technology has everything that CASE has ever wanted: dynamics for enterprise modeling, a higher level of abstraction for metadata, enforced reuse, the ability to handle unstructured data for storing design. The fact that traditional CASE vendors have finally awakened to this is a good sign. The recent joint venture between James Martin Associates and Intellicorp is a good example. The most exciting O-O CASE technology will not come from TI or KnowledgeWare, but from a new generation of companies with a fresh perspective.

HOTLINE: What role should MIS take in business reengineering?

KROL: MIS must study business reengineering principles to be able to translate them into new systems. The most important thing, however, is restoring a relationship with the business people. This relationship has been lost over many years of inbred mistrust. Rebuilding it will be key to successful business reengineering projects. Business reengineering is often a joint effort between MIS and the business community. MIS people are sharp and have good insights. They are also partly responsible for the mess we are currently in. However, if MIS continues having the same standoffish attitude that has characterized it in the past, the gap will only widen between them and their "business fundamentals."

HOTLINE: We're talking about developing new skill sets and exposure to business experiences that are not part of the traditional MIS professional's career path. If we've had trouble implementing CASE, what level of effort will be needed to bring us so much farther?

KROL: Again, from my perspective, it is not the magnitude of a technology effort, it is the *attitude* that needs to change—which is a lot harder. MIS careers should not have "digressions into business-related areas"; they should be intertwined with business.

HOTLINE: Assuming we can manage the changing role and perspective, what are the barriers to MIS undertaking business reengineering with object technology?

KROL: The barriers are the usual litany of things like a steep learning curve and lack of standards and expertise. The major impediment, however, is that MIS fails to recognize applications that would make objects in-

log maintenance, issue and receipt of components, cost allocation, etc. As IT moves toward software manufacturing and away from being a cottage industry, metrics will become ever more important.

BUSINESS OBJECT INTERFACES

An important component of competitive edge is to make employees more effective by widening the range of activities they are able to undertake. To enable this, employees need highly usable interfaces that:

- match their perception of the business world by continually reusing stable objects reflecting the business concepts that structure their business
- do not require the user to think in terms of computer applications
- enable the user to manipulate and assemble objects in response to whatever tasks or requirements arise

This is an attractive use of object orientation since it allows us to move forward and take new opportunities without major changes to our existing systems legacy. However, it has important IT consequences.

First, object orientation must be used to model the user's workplace and identify the stable business objects. This requires appropriate analysis techniques as well as raising the IT organisational issues of teams needing to focus around objects as apposed to applications. This cuts across the current applications-based IT development organisation.

Second, the business objects will only exist on the front end. Objects will need to draw on existing, host-based databases to access and store their instance data. A move to client/server architecture is therefore essential. It also raises the issue as of whether the client/server architecture itself has to be object based.

O-O gives IT the ability to leverage business worker productivity by providing tools that correspond to the concepts of the business environment itself. The only thing in a typical business that is not a business concept is the traditional IT "application." O-O passes flow of control to the end user, deemphasizing the large, monolithic application that structures and constrains what business workers can do. By increasing the correspondence between IT support and the business, we can eliminate much of the cross-correlation that goes on today when business people have to try to figure out, from the applications available to them, the answers to the questions and problems they are facing.

THE GROUP VIEW

From the OIG's perspective, O-O provides the opportunity to grow and extend today's legacy systems in exciting ways not possible with current, non-object technology. Furthermore, this can be done without discarding the enterprise-critical systems our businesses depend on right now. What is clear to us is the business needs of the future are about going out to where the business wants to go and rapidly integrating new opportunities into the business immediately. We can-

> ***Business is changing faster than we can afford to change the legacy systems—and faster than the legacy technology will allow us to make these changes.***

not afford to wait ages to modify today's legacy systems. We have to be able to quickly deliver new goods and services as the need arises. Business is changing faster than we can afford to change the legacy systems—and faster than the legacy technology will allow us to make these changes.

This is where the opportunity lies: modeling the workplace, delivering the support for new products and services, and integrating hitherto separate, large legacy systems. This points to a computing environment that blends mainframe-based legacy systems with distributed, opportunity-driven, front-end systems using O-O: an environment in which O-O is deployed to enhance and extend the business but is not the replace-all panacea.

Whilst short-term benefits can be gained from O-O, it is more important for companies to realise the long-term implications of O-O and plan accordingly. Management of this long-term strategy is likely to be a major challenge. ≡≡

References and suggested reading

1. Plant, N. A large-scale users' assessment of object orientation, Hotline on Object-Oriented Technology 3(5):1–6, 1992.
2. Object Orientation: An Assessment by Large Scale Users of Information Technology, a report of The Object Interest Group, February 1991.

Norman Plant is Chairman and a founder of The Object Interest Group, and a consultant specializing in the introduction of object technology into large IT organizations. He recently retired as a senior manager in the IT division of British Airways after 25 years service to the company. He can be reached at +44-252-836315, in the UK.

...continued from page 5

new technology will not be opposed by alternatives; it will be opposed by habit. It will be confronted by individuals who are disinterested, but who must understand and accept the terms of the new technology. The members of this larger community may actively oppose any change because they are weary of too many changes. For this reason, it is unlikely that the greater community will have many passionate advocates for still one more change, although it will still contain opponents. But, by far, the most common adversary will be habit and antipathy.

Once more, the advocate of the new technology must seek opportunities to integrate the changes offered by the technology with the established ways of doing things. The dimension of integration is in the arena of direct action. Most business actions are becoming more complex, and at the same time proportionally less training is being provided. If the new technology can be integrated within these action sequences in a dramatic and highly visible way, it is likely that the change being induced will be accepted. Personal information systems, information visualization systems, and systems that will simplify connection to other systems are examples of changes that can be readily integrated. Simple productivity improvements can be made more acceptable with individualistic learning systems grafted on to help the members of the community adapt to the new methods.

Regardless of the benefits offered or the tactics taken this battle will be a long one. In many ways it can be futile. If the technology is successfully introduced into the corporate culture, it will be one more contributor to the overall level of culture shock. Eventually, even this new technology will become a legacy system to be clutched as a talisman against the next new technology that survives its trial by combat.

Success in bringing object-oriented concepts to commercial data processing will depend on the skill with which the sponsor combats the challengers. Employing opportunities for integration with other technologies, integration with action patterns of the users, and integration within the total cultural context of the corporation will ease the battle considerably; it will also increase the ultimate value of the new technology. Despite integration, opponents, adherents, and circumstances will still create practical situations that will adulterate important, central concepts. In turn, these changes will produce new confrontations. If not handled effectively, this whole effort will do little more than make the next innovation harder to introduce and make the corporation a little more resistant to change. ≡≡

Reference

1. Musashi, M. *A Book of Five Rings* (translated by V. Harris), Overlook Press, 1974.

Frank Douglas is a New York-based consultant providing guidance in the dynamics of technological change, especially in database and rapid application development systems.

hotline on OBJECT-ORIENTED *technology*

Back issues available

Vol.3, No. 6/April '92 ≡ Thinking the unthinkable: reducing the risk of failure ≡ Mitigating madness with method: first establish what you value ≡ Championing object technology for career success in the 1990s ≡ Objects and actions in end-user documentation

Vol.3, No. 5/March '92 ≡ A large-scale users' assessment of object orientation ≡ Report on the Object-Oriented COBOL Task Group ≡ Interview with K.C. Branscomb

Vol.3, No. 4/February '92 ≡ The big prize: acceptance of O-O by the MIS community ≡ 1991—the year it all changed ≡ Making the transition to O-O technology

Vol.3, No. 3/January '92 ≡ Enterprise object modeling: knowing what we know ≡ Adopting objects: pitfalls

Vol.3, No. 2/December '91 ≡ Accepting object technology ≡ Adopting objects: a path ≡ Incorporating graphical content into multimedia presentations

Vol.3, No. 1/November '91 ≡ Leading the US semiconductor manufacturing industry toward an object-oriented technology standard ≡ Coping with complexity: OOPS and the economists' critique of central planning

Vol.2, No. 12/October '91 ≡ A modest survey of OOD approaches ≡ What is a "certified" object programmer? ≡ Perspective: investing in objects today

Vol.2, No. 11/September '91 ≡ From applications to frameworks ≡ Report on the 7th meeting of the O-O COBOL Talks Group ≡ Getting started with object technology: effectively planning for change

Vol.2, No. 10/August '91 ≡ Distributed object management: improving worker productivity ≡ Getting the best from objects: the experience of HP ≡ EC employs object technology for hospital information systems

Vol.2, No. 9/July '91 ≡ Multimedia is everywhere! ≡ Developing an object technology prototype ≡ Object-oriented capacity planning ≡ How OOP has changed our developmental lifecycle ≡ Modularization of the computer system

Vol.2, No. 8/June '91 ≡ Domain of objects: the Object Request Broker ≡ Object-based approach to user documentation

Vol.2, No.7/May '91 ≡ Hybrid object-oriented/functional decomposition for software engineering ≡ So, what makes object databases different? (Part 4) ≡ Using the generic application to solve similar domain problems ≡ Experiences using CLOS

Vol.2, No.6/Apr. '91 ≡ An artist's perspective of programming ≡ So, what makes object databases different? (Part 3) ≡ Moving from Pascal to C++, Part 3

Vol.2, No.5/Mar. '91 ≡ Large-scale object systems development ≡ Data definition languages ≡ Moving from Pascal to C++, Part 2

Vol.2, No.4/Feb. '91 ≡ Moving from Pascal to C++ ≡ So, what makes object databases different? ≡ Managing the evolution ≡ 1990: the year of realism ≡ Choosing a configuration for object development

Vol.2, No.3/Jan. '91 SPECIAL ISSUE: CASE AND METHODOLOGY ≡ Object orientation: stop, look and listen! ≡ The spiral model for object software development ≡ Bootstrapping object-oriented CASE ≡ A CASE tool for object software development

Vol.2, No.2/Dec. '90 ≡ Waiting for OODB2 ≡ Object modeling • *Singh* ≡ Object-oriented COBOL — the next generation

Vol.2, No.1/Nov. '90 ≡ Objects by teamwork ≡ Race for the database pennant: a third-generation manifesto

Vol.1, No.12/Oct. '90 ≡ Crown jewels: the value of OOT in MIS departments ≡ Why is an object database faster than a relational system? ≡ Visual programming object language environments

Vol.1, No.11/Sep. '90 ≡ Introducing OOPS through design — it's all in the words ≡ Design principles for modular product development ≡ Asset management and object-oriented technology

Vol.1, No.10/Aug. '90 ≡ WyCASH+: An application built within an OOP environment ≡ Organizing for reuse

Vol.1, No.9/July '90 ≡ Cognos and Eiffel: a cautionary tale ≡ What's in an object anyway? ≡ The impact of Windows 3.0 on the OOP market ≡ Introducing reuse into the development process

Vol.1, No.8/June '90 ≡ A taste of the USENIX C++ Conference ≡ Comparing relational and object database management systems part 2 — Composites, flexibility, and extensibility ≡ Case studies in prototyping, part 3 ≡ Object-oriented requirements analysis

Vol.1, No.7/May '90 ≡ Object-oriented development: not business as usual ≡ Comparing relational and object database management systems part 1 — Single paradigm programming is the key difference ≡ Case studies in prototyping, part 2 ≡ Understanding the recursive/parallel lifecycle

Vol.1, No.6/April '90 ≡ Lifecycle approaches ≡ A data manipulation language for C++ developers—a proposal ≡ Case studies in prototyping

Vol.1, No.5/March '90 ≡ Objects of desire ≡≡ Talking to project managers

Vol.1, No.4/Feb. '90 ≡ Greed: a complete example ≡ Talking to project managers ≡ OOP in the 1990s

Vol.1, No.3/Jan. '90 ≡ Adopting object-oriented techniques: management and organizational impacts ≡ Which language to choose, revisited

Vol.1, No.2/Dec. '89 ≡ C'ing into the future ≡ DATABASES: Standards for OODBMS: a manager's guide ≡ Synthesis: an object-oriented analysis and design method ≡ From languages to environments

Vol.1, No.1/Nov. '89 ≡ Which object-oriented language should we choose? ≡ An introduction to object-oriented database management systems ≡ Understanding object-oriented analysis

See back page for ordering information

INTERVIEW ≡≡

Enterprise reengineering with CASE and objects

Natasha Krol, interviewed by Robert Shelton

In this interview with industry analyst Natasha Krol of Meta Group, Applications Development Strategies, two trends stand out. First, our business systems must be rethought from the ground up. Second, although CASE may help MIS achieve technical reengineering of today's programs, objects will enable business process reengineering, which is essential for our companies to remain competitive. Ms. Krol suggests a new perspective that MIS needs to develop to cope with today's changing business and technology.

—Editor's Note

HOTLINE: Over the last several years, CASE vendors have pushed reengineering tools that claim to recover buried design treasure from today's applications; but when you use the term reengineering, you're clearly not talking about reprogramming, right?

KROL: That is correct. The term reengineering is used to describe two very different concepts. In the CASE context, reengineering means evolving, rather than replacing, old dysfunctional systems. It means salvaging whatever makes sense, and gradually evolving legacy systems into new architectures. In contrast, *business process reengineering*, as defined by Michael Hammer, means abandoning old systems in favor of radically reengineering the enterprise itself. It means recognizing and breaking away from old rules and outdated assumptions that underlie most companies' operations. Business process reengineering invites companies to free themselves from the bondage of COBOL and the iron grip of IMS, to rediscover the very essence of their business. It results in business leaders building new systems that reflect the new market realities they are facing today, instead of the realities and values of business 20-plus years ago.

HOTLINE: What makes object technology more applicable to business process reengineering than the procedures businesses have in place today, like Information Engineering (IE)?

KROL: Part of that has to do with current CASE. In today's CASE environments, the analyst still thinks in a specific solution-oriented manner. The current generation of CASE tools does not offer a compelling framework to take a deeper look at the business problem in its entirety. These tools do not motivate analysts to focus on the origins of the business problems themselves. Consequently, approaches like information engineering tend to mechanize or legislate the old ways of conducting business. Recent studies by Morgan Stanley point out that heavy investments in information technology have not contributed to significant increases in white-collar worker productivity because companies have been throwing sheer processing power at situations that require creative thinking and intelligence. Object technology is a better fit with business process reengineering than IE because going back to business fundamentals is consistent with the object-oriented paradigm. The process of building class hierarchies forces developers to address essential business principals as they create the basis for subsequent applications by building the foundation business classes.

HOTLINE: So you're saying that business reengineering involves truly understanding our businesses, instead of simply understanding the arbitrary processes and organization structures that happen to implement them today?

KROL: Yes. Business reengineering is about understanding the very fabric of the business, as opposed to the maze of disjointed applications that have been built over time, and applying this knowledge to the new environments. For example, when Brooklyn Union Gas rewrote the accounting portion of their customer information system they studied accounting textbooks and even talked to some college professors to make sure that their fundamental classes did not carry the baggage of their old processing applications. Truly accurate generic classes will enable more supple and malleable applications that are able to withstand change. These are the types of systems that will enable our companies to cope with the new market realities: a highly segmented customer base, nar-

Robert Shelton is the editor of the HOTLINE ON OBJECT-ORIENTED TECHNOLOGY. He can be reached at (415) 928-5842.

Natasha Krol is Program Director of Meta Group's Application Development Strategies service. She specializes in the practical applications of advanced technology, including object technology. Prior to joining Meta Group, Ms. Krol was Manager of Advanced Technology Research at Strategic Technologies Planning for John Hancock Financial Services. Previously, she analyzed Western technology at the Ministry of Foreign Trade in St. Petersburg, Russia. She can be reached at (415) 579-6615.