ATTENDANCE

	15 April	16 April
Dr. Walter S. Baer	Yes	Yes
Dr. Launor F. Carter	Late	Yes
Prof. Wesley A. Clark	Yes	Yes
Dr. Sidney Fernbach	Yes	Yes
Dr. Martin Greenberger	Yes	Yes
Mr. Jerrier A. Haddad	No	Yes
Mr. William Knox	Yes	Yes
Dr. J. C. R. Licklider	No	Yes
Mr. William L. Lurie	No	No
Dr. John R. Meyer	Yes	Yes
Prof. W. F. Miller	No	No
Mr. Roy Nutt	No	No
Prof. A. G. Oettinger	Yes	Yes
Mr. Kenneth Olsen	No	No
Dr. Alan J. Perlis	Yes	Yes
Dr. John R. Pierce	Yes	Yes
Prof. J. Barkley Rosser	Yes	Yes
Dr. Alan F. Westin	Yes	Yes
Dr. Ronald Wigington	Yes	Yes
Mr. Joel Cohen	No	No
Mr. John Griffith	Yes	Yes
Dr. Bernhard Romberg	Yes	Yes
Mr. Warren C. House	Yes	Yes

Exec. Session

COMPUTER SCIENCE AND ENGINEERING BOARD

AGENDA

Evening Session 15 April 1970

EXECUTIVE SESSION

The evening session of the Computer Science and Engineering Board Meeting will be held in Room 500A of the Joseph Henry Building and begins at 8:00 p.m.

The draft FCC Report (Mr. Billig, Chairman, FCC Panel) will be reviewed by the CS&E Board.

The Chairman of the Board, or Dr. Ling if present, will review development of day's activities of the Export Panel (Computer Technology Panel).

Day Session

COMPUTER SCIENCE AND ENGINEERING BOARD

AGENDA

Day Session 16 April 1970

Joseph Henry Building
Room 500A

0900 - 1100	Complete Review of FCC Report or start review of Education Report if FCC review has been completed
1100 - 1200	Dr. Holt, Information Centers for Decision Making
1200 - 1230	Lunch
1230 - 1300	Dr. Gilchrist, International Affairs
1300 - 1430	Computer Education Report Review
1430 - 1500	SIC Action Review
1500 - 1530	Computer Security and Files Management
1530 - 1700	Revised Board Structure Proposal

Perlis Report

THIRD DRAFT

THB:ls

3/27/70

COMPUTER SCIENCE EDUCATION

Report on a conference sponsored by the

Computer Science and Engineering Board

of the

National Academy of Sciences

under a grant from the

National Science Foundation

held in

Annapolis, Maryland

July 21 - 24, 1969

PREFACE

In July 1969, a group of approximately thirty experts from industry, government, and education met in Annapolis, Maryland to discuss the future of education in Computer Science. The chairman of this conference was Alan J. Perlis. In March 1970, a smaller group met in Washington D.C. to prepare this report which presents the findings and recommendations of the original conference. Final editing of the report was done by Thomas H. Bredt.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont)

1. INTRODUCTION

Scope of the Conference

A conference to study Computer Science Education in the United States was held in Annapolis, Maryland, in July, 1969. The conference was sponsored by the National Academy of Sciences' Computer Science and Engineering Board under a grant from the National Science Foundation. The conferees attempted to identify both the proper goals for a college or university education in Computer Science and, in a broad sense, the routes and structures for achieving those goals. Discussion and recommendations focused on the estimated 20% of the computer field personnel who, even in the near future, must be college trained. However, the bulk of the people working in computing for the next 5 to 10 years will not be four-year college graduates but will, rather, receive their final training in high schools or two-year colleges. This meeting did not address the critical problem of training such people. A conference on this subject should be organized.

A Profession or a Science?

A recurrent question that threaded its way through most of our discussions was whether "Computer Science" ought to be a "professional" or "scientific" study. The issue, which in the end proved to be somewhat vacuous, was whether the graduate was to go on to "design things" (like the engineer) or "illuminate truth" (like the mathematician). The conclusion of the group was that a variety of graduates is necessary and should be produced but that the distinction in their education should be achieved by the extent and, therefore, the depth and richness of their education in Computer Science and not by "separate tracking" or education in different disciplines. Computer Science, therefore, is a discipline which has both practitioners and scholars.

The Numbers Problem

The first issue addressed was the numbers problem: How many and what kind of people do we need to educate? Two approaches were used in attacking this problem:

(1) extrapolation of equipment-support requirements.

(2) reasoning by analogy with other fields.

The first approach assumed the existence of about 10,000 computers in the United States in the 1975-1980 period (these are machines that need the support of computer professionals). A further assumption was that the number of computers and the staff needed to support them would "plateau" thereafter. This provided a base for the manpower computation.

With due consideration for the "mix" of large, medium and small installations, a support group of about 600,000 professionals was deemed necessary. The long term, steady-state condition (about 30 years from now) assumed that these people will all be college trained. Then, assuming a working life of 30 years, the replacement rate will be about 20,000 per year.

The current college-educated "professional" population in the computer industry is believed to be about 100,000. This number is so far below that needed to competently staff the nation's computer installations that even a 30,000 per year influx of trained people would be desirable. Such a "production" rate, of course, is not currently possible. This number is presented only to support our contention that a 20,000 per year rate is a reasonable national goal.

The second approach to the "numbers game" adopted was to estimate the support population in relation to other, better understood and more mature, disciplines. Compared to the 40,000 engineering graduates per year (from all engineering disciplines) our chosen target of 20,000 per year seems reasonable.

Compared to the 10,000 medical doctors per year and the 20,000 nurses per year, the anticipated rate retains its plausibility.

We, of course, understand the fatuity of attempting to make these kind of predictions for periods beyond the next five years. Our chief concern is that we are not overstating the need. We believe we are not.

The Mix of Graduates

Twenty thousand graduates per year is the goal. What kind of training should they have? What is the "mix" of degrees? These were the next issues addressed.

Many factors influenced the assessment of the proper proportions. In net, it is the judgment that the 20,000 per year should be broken down as follows:

500	PhD's/year
3,500	Master's/year
16,000	Bachelor's/year

The Educational Path

As mentioned above, Computer Science is seen as a single, coherent academic discipline. We reject the notion that "theoretical" Computer Science and "practical" Computer Science are so different that they cannot share the same base. For that reason we recommend that there be, in any university, a single Computer Science "track". An undergraduate "core" curriculum (with electives) will produce a bachelor-level graduate who has a thorough grounding in the fundamentals of his field. Graduate study will lead, inevitably, to deeper understanding and greater accomplishment.

Single-tracking the Computer Science student creates a number of serious problems as to the content of the core curriculum. We understand the complexity of the issues raised and do not, here, propose or recommend any curriculum. We are convinced, however, that the benefit of not having a splintered discipline far outweighs the disadvantages induced by the necessities of accomodation.

The Neglected Majority

An important though paradoxical, result of our deliberations is what we might call the plight of the neglected majority. At the present time, about 80 percent of the computing done in the United States is in support of business applications. Programming in COBOL, business data processing systems design, development, and operation represent the sole activity of the majority of the people in the computer field today. These kinds of activities, similarly, will represent the major involvement of people during the projection period.

Why not, then, direct our educational policies and courses towards the more specialized needs of this group? The arguments come, in essence, to the notion that the data processing community benefits from students educated in the fundamentals of Computer Science.

Costs of Educational Computing in Computer Science

Except for the cost of providing computer services, the cost of educating the Computer Science student is not significantly different from that of educating the Physics or Chemisty student (again without laboratory expenses). However, an important, but costly, part of the student's training is the provision of experience with a computer (or with computer services). This necessary laboratory experience is estimated to cost:

$ 20 per man per year	for the non-Computer Science student
$1,000 per man per year	for the Computer Science BS years
$2,000 per man per year	for the Computer Science MS years
$4,000 per man per year	for the Computer Science PhD years

This represents a cost of about $90 million per year for Computer Science education.

2. RECOMMENDATIONS

1. At the present time there exists a recognized shortage of professionals trained in Computer Science. This shortage is felt in all areas where computers are used. To remedy this shortage, at a fairly expert level, we recommend the establishment of strong master's programs in Computer Science in degree granting institutions.

By a strong master's program we mean a program that will provide a sufficient education for those professionals who are going to fill the need for trained practitioners of Computer Science in industry and government and who will improve the efficiency and scope of computer operations.

Furthermore, we recommend that master's programs in Computer Science contain strong elements of laboratory training in the development and utilization of computer systems.

2. To remedy the above mentioned shortage at a less expert level, we recommend the establishment of strong bachelor's programs in Computer Science in degree granting institutions.

By a strong bachelor's program, we mean a program which will prepare students for employment as working computer professionals and for advanced education in either or both master's and PhD programs in Computer Science.

Furthermore, we recommend that bachelor's programs in Computer Science contain strong elements of laboratory training in the development and utilization of computer systems.

3. We recommend that the development of doctoral programs in Computer Science continue at its present rate. While these programs have done well, it is recommended that they continue to be supported by (1) graduate teaching and research fellowships, (2) post-doctoral teaching fellowships to aid in acquisition of new faculty, and (3) support of new and different computer facilities. Examples of these new and different facilities are satellite computers, processors for film and TV animation for instructional purposes, hybrid computers, converters to and from other systems, and advanced equipment such as is developed as a result of the investment of national resources in research and development programs for defense, space and other sciences.

4. It will be essential for the universities and colleges to greatly expand their students' opportunities to learn the essentials and principles of all elements from problem formulation to computing realization and for these institutions to be aware of the part that Computer Science can play in this expansion. It is recommended that support be provided to implement cooperation between computer scientists and individuals of other departments to the end that individuals from other departments be encouraged and supported in providing opportunities for students to gain insight and knowledge in part or all of Computer Science. All reasonable efforts should be made to encourage interdepartmental cooperation in this area. Finally, both research in the general area of application and materials preparation directed toward teaching deserves support, especially when each is planned to support the other.

5. It is recognized that the need for professionals in Computer Science is a national one and, therefore, all effort should be made to provide support for the development of bachelor's and master's programs with the widest possible geographical distribution.

6. In the rapidly changing field of Computer Science and computer related activities, up-to-date information on research is needed and is hard to get. Under NSF sponsorship, the Southern Regional Education Board has prepared surveys of college and university educational activity in the computing sciences, but apparently no agency is doing anything similar for research in this field. At the same time, graduate departments have a great need for, but possess very little information on what research in computing sciences is being sponsored; who does the research, who sponsors it, and at what levels.

In a relatively stable field like mathematics, a strong need has been felt for up-to-date information about the nature of education and research in the field, and the amounts and sources of its funding. These needs resulted in the NSF-sponsored Survey of Research Potential and Training in the Mathematical Sciences (c. 1957), and the reports of the Ford Foundation-sponsored Survey Committee of the Conference Board of the Mathematical Sciences (c. 1967). The latter committee apparently will maintain a continuous inventory from now on.

We recommend that support be provided for a continuing research and manpower committee whose mission would be to maintain a continuing national inventory of research activity and manpower needs in Computer Science.

7. In the current environment, there exists a large and growing number of highly trained and competent PhD's from related fields. Many of these people would like to redirect their talents to Computer Science. We, therefore, recommend that special attention be paid to the invaluable opportunity for creating applications programmers, systems programmers, and Computer Science faculty and research persons by retreading recent PhD's in other fields. In particular, we recommend the institution of transdoctural programs that will complete the training of such people in one to two years.

3. QUESTIONS

The conference attempted to answer the following four questions:

(1) What are the manpower needs in Computer Science and what resources will be required to meet them?

(2) What educational programs should be implemented to produce the computer scientists in the needed kinds and numbers?

(3) How does Computer Science relate to the use of computers in other activities?

(4) Should there be separate degree programs in software engineering?

Findings were obtained that led to answers to these four questions. These findings are presented in the following section. The answers to the questions are embodied in the recommendations of the conference given in the previous section.

4. FINDINGS

Question 1: Manpower and Resource Needs

During the last decade Computer Science has been recognized as a separate and respectable academic discipline and has made considerable progress in this period. The first graduate programs in Computer Science were initiated some ten years ago and by the mid-sixties there were several Computer Science departments which offered the PhD degree. Since then, many more departments and Computer Science programs have been established and today there are more than thirty graduate programs offering advanced degrees in Computer Science.

The progress made during this decade is very impressive and the overall quality of most of these programs is quite high. The quality of the graduate students entering Computer Science has also improved dramatically over the last five years and it is now clear that Computer Science is attracting some of the brightest young people which is already reflected in recent Computer Science PhD's.

It appears that during 1970 some 200 PhD's will be awarded in Computer Science and that the existing programs are capable of producing some 250 PhD's. After several different ways of estimating the demand we believe that in the "steady-state" we will need to graduate some 400 to 800 Computer Science PhD's per year. This leads us to conclude, assuming the lower estimate, that the PhD output should be increased at a moderate rate to reach the steady state figure. Thus, the existing PhD programs should be strengthened and expanded and new ones should be created selectively. On the other hand, no crash program seems justified at this time. It should be kept in mind that these findings are based on very rough estimates and they should be refined as more information becomes available.

We now outline some of our calculations which lead us to these findings:

1. It has been said that there are about 67,000 computers in the United States in 1969. Let us assume the following distribution of sizes of installations and staff.

Size of Installation	Large	Medium	Small
Number of this Size	1000	10,000	56,000
Avg. CS Employees per Inst.	100	30	3
Avg. No. of PhD's per Inst.	5	1	0

Then the desired total number of Computer Science employees is 568,000 = 1000 X 100 + 10,000 X 30 + 56,000 X 3 and the desired total number of Computer Science PhD's is 15,000. If we assume that the productive life-time of a PhD is thirty years, we conclude that we need $\frac{15,000}{30}$ = 500 Computer Science PhD's per year. There will have to be about 8,000-9,000 Computer Science PhD's in the universities to meet the computer need and to maintain the existing faculties. The calculation of this estimate follows. To satisfy this need, we require $\frac{9000}{30}$ = 300 PhD's per year. Thus, the total demand is for about 800 PhD's in Computer Science per year. We belive that this estimate is high.

2. The number of PhD's required to teach Computer Science courses can be estimated as follows. The term "unwashed" refers to students taking Computer Science courses who are not Computer Science majors. There are about 6,000,000 students enrolled in four year colleges and we assume that each of them will take at least one Computer Science course during their undergraduate years.

a. Unwashed $\frac{6{,}000{,}000}{8 \text{ sem} \times 150/\text{class}}$ = 5000 classes/year

b. BS $\frac{2 \times 15{,}000 \times 6 \text{ classes/yr}}{30 \text{ stud/class}}$ = 6000 classes/year

c. MS $\frac{2 \times 3500 \times 8 \text{ classes/year}}{20 \text{ stud/class}}$ = 2800 classes/year

d. PhD $\frac{2 \times 500 \times 6 \text{ classes/year}}{20 \text{ stud/class}}$ = 300 classes/year

<u>The total number of classes per year is 14,000</u>. Assuming each faculty member teaches two courses per year, 7,000 faculty are required in Computer Science.

3. We next compared Computer Science with mathematics. Currently, mathematics is producing about 1000 PhD's per year and most of them are absorbed (with great difficulty) by universities and colleges. If we estimate that in the future two to three times more students will be taking mathematics courses rather than Computer Science courses, we conclude that we should be producing 300-400 Computer Science PhD's per year to maintain the Computer Science faculty. This would suggest that the average production of PhD's should be from 600-800 per year.

4. We calculate the yearly dollar cost to educate all students in Computer Science as follows.

a.	Unwashed hours = (100 batch)	\$10/year	(6,000,000/4) 1.5M	\$ 15M
b.	Bachelor (CS) hours = 25_b + $(1 - \alpha)100_t$	\$1000/yr	(3 x 15K) 45K	\$ 45M
c.	Master (CS) hours = 10_b = $(1 - \alpha)200_t$	\$2000/yr	(2 x 3500) 7K	\$ 14M
d.	PhD (CS) hours = 20 machine	\$4000/yr	.5K $(\frac{500}{2} \times 2)$	2M
				\$ 76M

Question 2: Educational Programs

A Coherent Discipline. We view Computer Science as a coherent academic discipline. The educated Computer Scientists will be trained in the design, analysis and construction of computer systems---complex mixtures of both hardware and software. We believe that there is a core of knowledge fundamental to the undergraduate's education and independent of his future course of study. We find no compelling reasons that lead us to suggest that Computer Science is appropriately placed within any particular classical academic department or college. Our strong concern is that in a given university, there be only one undergraduate program concerned with the science and engineering of computing (a student wishing to enter Computer Science from an "adjacent" field will have the traditional academic remedy of "making up" the necessary prerequisites).

There will, of course, be programs of study within other departments which concern themselves with the study and use of computers from differing perspectives.

System Design. Much of the programming activity in the country is aimed toward the construction of large and complex systems where most of the difficulties arise from the size and complexity of the systems and the fact that the systems are usually poorly or vaguely specified at the outset and even during the progress of their development. These problems fall into a category that represents an important area concurrent to and perhaps a part of Computer Science. The phrase " software engineering"has been proposed for the study in and of this technology. Some of what is known as "systems engineering" or "operations research" falls directly into this problem area. It is in the area of design and development of large computer programs for such large systems that there appears to be a lack of organized instruction in higher education, here or anywhere, at the present time.

Systems Laboratories. We consider the laboratory-experimental aspect of the training of students in Computer Science to be vital to their development. We therefore believe that the establishment of computer systems laboratories is an important part of the curriculum of both undergraduates and graduates in Computer Science.

There are many substitute plans that could conceivably serve to fulfill the same purpose as the computer systems research laboratories, e.g., summer employment in industry, cooperative work projects with industry, or part-time employment in a computation center on campus.

We believe that a team of six students can be given a very significant experience for $1,000.00 per student or $6,000.00 for the whole team for a one-quarter laboratory.

The Master's Degree Program. The nation has need for people to do a variety of jobs connected with computers. A substantial number of these people will be involved with the design and implementation of large computer systems each consisting of an assembledge of equipment (hardware) and a complementary collection of systems and library programs (software).* Those people involved in such design and implementation activities are carrying on a profession which is in a very real sense similar to that pursued by professional engineers. It is our belief that professional educational programs

---

* time-sharing systems, traffic control systems, command and control systems, management information systems, etc.

should be available which are specifically planned to provide the knowledge necessary to carry out these computer system design activities. By analogy with the engineering situation, it seems clear that these educational programs should be at the graduate level, leading to a master's degree; that they should build on a relevant bachelor's level education; they should be specifically planned as terminal, professional master's programs; and that they should consist of courses at the level of scientific generality offered to beginning doctoral candidates, i.e., should not be vocational type courses. Initially, the bachelor's level education of those entering this master's program will probably consist of a degree in engineering, physics, mathematics, etc. with a minor in Computer Science. As the number of Computer Science baccalaureates increases, a larger proportion of the students entering the master's program will have a deeper preparation in Computer Science that will bring about improvements in the quality of the program.

It is recognized that there may be a master's program in other academic areas which accent computers and their applications. It is felt that these programs should be designed by and largely be manned from within these academic areas. The Computer Science faculty should be used to teach the Computer Science courses included in these applied programs.

Question 3: The Relation of Computer Science to the Users of Computers

We find that there are a wide variety of problem areas that will benefit from advances in Computer Science. Future research is certain to expand the areas where computing can be effectively used. By no means will all of the problem areas lie in the field called Computer Science.

Problem Formulation and Matching: A Vital Segment. If we are to realize the potentialities of computing systems at a reasonable rate, we must look forward to the education and development of men and women across a very broad spectrum. It is easy to recognize the inevitable needs for certain kinds of people, such as:

- researchers into the understanding and expansion of what algorithms and computing systems can do.
- systems programmers competent to guide, lead, and do the development of major software systems.
- operators and routine programmers to run tens of thousands of installations.

As we attend to such clearly recognized needs, and, as well, to such crucial needs as increasingly effective attention to "wholeware" -- to the hardware and software of a computing system as a whole -- planned together as well as working together, we must not forget the vital segment of the spectrum associated with matching the problem to the computing system.

Problems do not arise in forms suitable for attack by computing systems. Those that seem to us "just made for a computer" came to that state by much human effort. If we are to tackle new problems -- or new versions of old problems --

effectively, bravely, pioneeringly, and successfully, it will be because individuals or small groups have done a good job of problem formulation, and because individuals or small groups have used the available computing systems and applications programs to deal effectively with these well and carefully formulated problems.

Neither phase of this task can be done wholly alone:

- Problem formulation often requires both repeated trial and exploration and insightful understanding of what computing facilities are really at hand.
- Bringing a good foundation to successful computing often requires guidance, sometimes repeatedly, from a version of the problem more true to life than the given formulation.

It will be essential for the universities and colleges to greatly expand all student's opportunities to learn the essentials and principles of all elements from problem formulation to computing realization.

Where a department of Computer Science wishes to lead in offering such opportunities, or to cooperate in offering them, that department should be especially encouraged and supported.

We feel it would be quite unrealistic to expect all departments of Computer Science to commit significant resources to this problem (indeed, there may prove to be no one area to which all these departments will, or should attempt significant contributions). The need is large, all who can and would should help to shoulder the burden.

Other departments with competent and interested staff should be encouraged and supported in providing opportunities for students to gain insight and knowledge in parts of all of this area. All reasonable efforts should be made to encourage interdepartmental cooperation and co-working.

If opportunities are to become widely available, there will have to be significant investments of time and efforts to develop materials ranging from case studies to organized presentations. Research into the credentials of how these problems are effectively formulated and brought to computation can and should have relation to mutual support with the efforts to develop materials.

Both research and materials preparation deserve support, especially when each is planned to support . the other.

Question 4: Software Engineering

Finding: "Software Engineering" is not a good phrase to identify an academic discipline and its use for that purpose should be discouraged. The reasons are as follows:

Hardware and software are intimately related. Ten years from now many functions that are now handled by software will be either hardware functions or shared hardware-software functions. The term "software engineering" emphasizes the distinction. It is very important to emphasize the interrelationships. "Computer Science" is a far better term for this since its programs include the subjects and views of "software engineering."

"Software Engineering" may, however, be an apt description of some activities currently being studied and taught within Computer Science.

5. SUPPORT

The documents in this section support the recommendations and findings presented in preceding sections.

What Kinds of Computer People are Needed? (B. Gilchrist, S. Moore, J.W.Graham)

Exclusive of installations involving special purpose equipment, or equipment for specific special purposes (e.g., process control), as well as those involving very small machines, there are on the order of 25,000 installations in this country. Very roughly, they are organized by size and purpose like this:

	Number of installations	Type of installation	
		Scientific	Commercial
Large	1000:	800	200
Medium	10,000:	5000	5000
Small	14,000:	4000	10,000

By large we mean the class of computers of the IBM 7090 type and their third generation successors such as the UNIVAC 1108, CDC 6600, IBM 360/50, 65, 67, 75, etc., GE 635 and 645, PDP 10 and Sigma 7. By medium we mean the class of computers such as the B5000, GE 235, IBM 360/40 and 44, CDC 3300 and 3400, and the Sigma 5. By small we mean the class of computers such as the PDP 8 and 9, HP2000A, IBM 1130, 1800 and 360/30, Honeywell, etc. An installation is a computer and the environment in which it is serviced, i.e., the direct support personnel required to maintain the computer's flow of input and output.

For purposes of designing college-level training and education programs, the group of people associated with the operation of small installations must be disregarded. That group (typically users of 360/20's) has as great a need as the others for competent people, but unfortunately the needs of larger installations for trained people deplete the smaller installations of whatever talent is available. By default, that section of the computer world becomes staffed by poorly trained people.

For the groups associated with medium and large installations, the people to be trained fall into these groupings:

Researchers

Systems Analysts

Systems Programmers

Applications Programmers

User/Programmers

Users

Researchers identify new things to compute, new methods to compute that which we have been computing and reorganization of the patterns of our thought by which we attach significance to the results of computation.

Systems analysts map a pattern of data production and data use into a sequential organization of data acquisition, transformation and production based on mechanical methods and coded representations. Naturally because of volume and traffic rate the computer plays a central role in this organization.

For systems programmers a distinction between two types was made. Some qualify as "advanced" systems programmers, e.g., they can modify OS360 to suit their firm's needs; others not so qualified can detect trouble, perhaps even identify its source in the operating system and know to whom to turn for help in fixing it. The group labeled "users" are those who merely know whether or not the results are correct.

A different breakdown of people needs is given in the following outline:

RESEARCH

Devises new tools and applications.

Needs specialists in (hardware
(software
(combination of the two

DEVELOPMENT

Develops those tools and learns how to use them.

Also needs experts in (hardware
(software
(combination of the two

APPLICATIONS

Requirements (or results) which have to do with data.

How to (select) a system.
(use)

How to measure effectiveness.

OVERALL

And all three groups deal with both theory and practical application.

Educational programs in Computer Science are concerned with producing people educated and competent in research, development and applications. The output of such programs, however, is negligible in number compared to the number employed in industry in activities said to be research, development and applications. As an industry (the committee agreed that it is not a profession), some 500,000 people are engaged more or less full time, but it has an abnormally high proportion of very incompetent people.

By incompetent is meant:

If one takes as a measure of competance (a) The number of "lines" of debugged code written per day, and/or (b) The computer space and time required for the execution of debugged programs -- one finds that an enormous range exists in the industry over comparable tasks. We quote some figures supplied by SDC (Communications of the ACM 11 (1968), 6). Performance by 12 programmers with 2 and 11 years experience on a specific logic problem:

Performance on	Worst/best
Debug time used	26/1
Computer time used	11/1
Coding time	25/1
Code size	5/1
Running time	13/1

While this is a small sample and further studies of larger populations should be made, many managers believe that the dispersion reflects the state of relative competence in the industry.

Attention was turned to a different view again: what sort of person does an employer look for and hire? We listed these specifications:

1) A certain gleam in the eye, vaguely defined as motivation.
2) Some knowledge of the mechanics of computing; e.g., the applicant has run some computer programs.
3) Problem solving adaptibility.
4) Communications skills (in both directions)
5) Ability to be self-critical.
6) Elementary knowledge of statistics (this last is weak, or optional)

Resources (A.J. Perlis, J. Giese, B. Gilchrist, J.W. Graham, J. Rowe)

We have a number of figures and tables which might be of interest. In education, the University of Waterloo has chosen to commence with the Bachelor of Science program in Computer Science and to develop from that upward to the MS and PhD programs. In the United States development in the opposite direction has generally been followed. It is recognized by Waterloo that the first approach is a somewhat more difficult path to follow, it being more difficult to upgrade a bachelor's program than to downgrade a PhD program.

This committee strongly feels, and this is the first recommendation, that major educational efforts should be spent in the development of Bachelor of Science programs in Computer Science in the USA over the next few years. Furthermore, the committee concurs with the Waterloo experience that the program should include significant amounts of practical, hands-on experience with real computer systems problems. Hence the committee feels, and this is a second recommendation, the BS program will be greatly aided by and should include laboratory courses and/or cooperative ventures with industry and government during the school semesters or over summer periods. The committee does not feel that the development of MS programs has the same priority as the two extremes, BS and PhD. Indeed, the MS program contains material only superficially different from the BS program and serves mostly as a springboard for those switching fields and as consolation prizes for those unable to complete PhD programs. The committee next considered the needs of the non-computer scientist being educated in the universities, since it became clear it would not be feasible to educate as many specialists as one might need in this field in the next 10 years.

The first calculation we made we call the Waterloo computation. At Waterloo there is an IBM 360/75, costing 125K per month. Student jobs account for 1/10 of the system time on that machine or if you will costing about 12.5K per month. Considering cost in the support or overhead equal to that of hardware we have a cost of $25,000 a month for student jobs. For that cost the productivity is 5,000 runs a day or 100,000 runs per month. Considering a productivity of four cracks at the machine per problem, this means that that system is capable of absorbing 25,000 problems per month. Consequently, given a student population size and a number of problems one can come up with various estimates as to what it costs to provide undergraduate computer experience for the non-computing specialist, i.e., someone who does problems of a relatively small size. We came up with one figure assuming 25,000 students in the university of one dollar per problem per student per month. The size of those problems is such that their programs are limited to one second of cpu time and the students are not charged for disc file time but they generally do not include much file work.

Over a ten month academic year a system of this kind could support students giving them 10 problems over an academic year at a cost of 10 dollars per student per year in a 25,000 student population which almost reaches the student population of the largest universities we have in the United States today. Now this figure is substantially below the figure in the Pierce report which runs closer to 50 or 60 dollars per year. That means if we wish to attain the Pierce report figure we could have the student doing 50 problems per year, which is probably much too heavy a load for non-computing specialists!

Now this leads us to make a third recommendation. We recommend that funds be made available so that a cost analysis study can be made of the specification and use of various systems for handling bulk student jobs for the non-computing specialist at different student population levels. It would be hoped to provide a study that would say - at the cost level at which we have spoken, given a student population of 1,000, system A would provide computation at the rate of $50 per year at a level of between 10 and 50 jobs or problems per year. At a student population of 5,000 system B will similarly provide, at 10,000 system C, at 30,000 system D. Such a specification of systems is not now available to the educational community. Of course, these systems need not be unique. There can be many systems in each of these categories. Neverthelss, it is the feeling that at all four of these student population levels, 1,000, 5,000, 10,000 and 30,000,systems can be found which are of economical comparison to the Waterloo system.

We arrived at an estimate that to turn out 300 PhD's per year in Computer Science, we were talking about an estimated machine cost of $9 million a year. This is the machine cost required to support PhD theses and PhD education at the level of 300 PhD's per year. Thus: to produce 300 PhD's per year it is estimated that it will take 30,000 dollars per PhD in machine time or a total of 9 million dollars in machine time for the PhD production of 300 PhD's.

For the Bachelor of Science program in Computer Science, assuming six courses in their program that are in the field of Computer Science, thus not counting auxiliary courses, and an education program that will turn out 15,000 BS Computer Science students per year, a figure of 15 million dollars per year in computer time was arrived at.

For the Master of Science program, a figure of 5 million dollars per year in hardware costs was obtained.

The total cost in hardware is 29 million dollars per year. One of the figures that we used was that the EDP industry would be taking in about 100,000 people per year. What percentage of these should be PhD's? Figuring that one percent should be PhD's we get a desirability of producing a thousand PhD's a year. Our feeling on the matter was that by 1975 we might be able to produce 1000 PhD's in Computer Science, but that we would not be able to produce 1000 PhD's per year by 1975. If you can get up to about 300 by 1975 this would be about what we could expect. It seems to double about every two years.

From whence comes this figure of 15,000 BS students per year? Is it attainable? At the present time in engineering and mathematics the output per year is of the order of 50,000. Now assuming there is no major change in size of total undergraduate enrollment in engineering and science schools but that quality Computer Science undergraduate programs do come into being, how many of the 50,000 per year could we expect to prefer an education in Computer Science? We believe that without a great deal of heavy advertising or pressure of any sort, 20-30% of the undergraduate enrollment in mathematics and engineering programs would shift into Computer Science programs, if there were existing quality undergraduate programs in Computer Science. Furthermore, the percentage is probably conservative. That means of the 100,000 per year that are required in the EDP area, 85,000 are probably going to have to remain or be non-Computer Science baccalaureates. We also made an estimate of Computer

Science faculty costs and came up with an estimate of 45 million dollars per year for that part of Computer Science faculty costs devoted to Computer Science education alone at the three levels being well aware, of course, that there are other costs associated with their education outside the Computer Science Department. But we are talking now about cost of a faculty of about 1500. Waterloo argues that they are producing 200 Baccalaureates per year to service 1,000 computers in the province of Ontario. There are 67,000 computers in the USA. Consequently, if we assume that the ratios are comparable, this leads to 13,400 output in the USA to service these computers, if we adopt that ratio. This compares reasonably well with out 18,000 figure.

The following set of figures, arrived at differently from the figures just cited, tend to corroborate this level by about 1975. A conservative estimate of the prospective demand for the products of the Computer Science educational system.

1. In the long run the overwhelming majority of Computer Science graduates at all degree levels will go to non-academic employment. For the estimates we shall make later, we shall need to estimate the number of "Computer Science" positions which should be filled with Computer Science trained people if possible at computer installations in the U.S.

   a. It has been said that there are about 67,000 computers in the U.S. in 1969.

   b. Let us <u>assume</u> the following distribution of sizes of installations and staff.

SIZE OF INSTALLATION	LARGE	MEDIUM	SMALL
NUMBER OF THIS SIZE	1000	10,000	56,000
AV. CS EMPLOYEES PER INST.	100	30	3
AV. NO. OF PhD's PER INST.	5	1	0

Then the desired number of TOTAL "CS" EMPLOYEES
1000 X 100 + 10,000 X 30 + 56,000 X 3 = 568,000
and the desired number of TOTAL "CS" PhD's = 15,000

c. These positions are not now filled by Computer Science graduates. We assume it would be desirable to replace them gradually by Computer Science graduates to upgrade the computing profession.

2. Let us assume that the computing profession remains static at about this level, i.e., that increases in efficiency make new people available for an inexhaustible set of new problems. Let us assume that we have a rather rigid slowly varying working population, like the Civil Service. This may not be too unreasonable to assume, since these professionals might become union-organized (as teachers are now). If we assume a working life of about thirty years, then in the steady state we shall have to replace about $\frac{568,000}{30}$ = 19,000 Computer Science employees per year and about $\frac{15,000}{30}$ = 500 Computer Science PhD's per year.

3. Composition of 19,000 Computer Science graduates.

If we assume that about 20% of these graduates seek advanced degrees, this means about 4,000 advanced Computer Science degrees per year. If we claim 500 PhD's per year, this leads to a need for

500 PhD's
3,500 Masters per year
15,000 Bachelors

in the Computer Science area.

4. Conservatism of this estimate.

   a. The assumed static "CS" employee pool is about $\frac{500,000}{200,000,000} = 0.25\%$ of the total U.S. population.

   b. 19,000 graduates per year is about half the number of engineering grads (40,000) per year. That doesn't sound unreasonable. Computer technology should be about as widely appliable as engineering.

   c. For comparative purposes consider the fraction of our manpower resources devoted to medicine and associated subjects. We produce about 9,000 physicians per year. They must be backed up or supplemented by about 18,000 nurses, technicians, dentists, and various forms of physiologists, etc. As a guess, about 27,000 graduates per year are devoted to problems to health.

      You might argue that since medicine absorbs a fairly small fraction of our economic output, and since computing is (or will be) involved in all of man's activities, including medicine, perhaps the output of Computer Science graduates could safely be increased to the level of medicine (and associated graduates) or 27,000 eventually.

   d. Some Computer Science enthusiasts assert that the growth of Computer Science may be 100,000 per year.

      In a steady state process, with a thirty-year working life, this would lead to a CS employee-pool of

$$30 \times 100,000 = 3,000,000$$

If the population of the U.S. remains static at 200,000,000, this would mean that the pool would contain about 1.5% of our population.

5. Nothing has been said about the provision of refresher courses for the people in the pool who will constantly become obsolete. If you provided a "refresher" or updating course once every five years, this comes to 0.2 course (three weeks?) per year. Even if you restricted this updating to the lucky employees at the large and medium installations, somebody would have to provide about

$$0.2 \times 400,000 = 80,000$$

student courses/year. Even if these courses operate at 100 students per section, you would have to run about 800 refresher course-sections per year.

If we are less generous and send only 10% of the pool to refreshers, this cuts the total to 80 course sections per year. That ought to be a tolerable burden for the educational system.

6. Nothing has been said about providing computer "service" courses for non-Computer Science students.

It may seem ridiculous to staff the small installations with graduates. To handle this we suggest that we reinterpret our imagined program. Let us say that we provide instruction and facilities to produce 190,000 graduates per year. If about a third of these drop out after the first two or three years, they would probably have to be content to work at the small institutions. Actual graduates go to medium or large places. We would assume that the computer industry would be included as part of the large installations.

One final point. The figure of 15,000 baccalaureates is considerably lower than we would like. Arguing about 100,000 entries into the EDP area a year are needed, we figure that 25,000 come from business schools and industrial administration programs, 25,000 by upgrading from their current positions. This leaves 50,000 coming from colleges, and we are only providing 1/3 of that. That means that 35,000 are going to come from a lower educational level than baccalaureate Computer Science programs. One of the consequences of providing 15,000 baccalaureates in Computer Science will be a temporary diminution of the number of people needed in the field. But we all agreed that this diminution would be temporary. The more trained people that you have presumable the less total number you need. However, we would really prefer that all 100K came out of baccalaureate programs in Computer Science.

We merely want to point out that the figure of 15,000 per year is, in our judgment attainable right now if baccalaureate programs are introduced.

Bachelor's Degree Programs in the Computer Sciences (J.W. Hamblen)

History and Status to June 1969. From 1960 through 1968 discussions relative to the desirability of and content of an undergraduate degree program in the Computer Sciences centered about the activities of the Curriculum Committee on Computer Sciences of the Association for Computing Machinery. Two reports were published which have had considerable impact upon the structure and content of bachelor's degree programs in Computer Science. The first report was published in the May, 1964 issue of the Communications of the ACM and was called "Preliminary Recommendations for an Undergraduate (Bachelor's) Degree Program in Computer Science." The later report appeared in the March issue of the Communications and was entitled "Curriculum 68: Recommendations for Academic Programs in Computer Science."

By June 30, 1965 it was estimated that there were 11 bachelor degree programs in Computer Science in operation and an additional 81 programs were expected to be placed into operation by 1967-68. [1.]

By June 30, 1967 30 bachelor's degree programs in Computer Science were reported in operation with 1727 majors and 175 graduates for 1966-67. [2.] In addition, another 53 bachelor programs were reported going with nearly 3000 enrollees and 300 graduates. The latter included Data Processing, Information Sciences, Computer Science Options in various academic areas the largest contributors being mathematics and electrical engineering.

----

1. Hamblen, John W.
   Computers in Higher Education: Expenditures, Sources of Funds, and Utilization for Research and Distribution 1964-65 with Projections for 1968-69, Southern Regional Education Board, (Atlanta), 1967
2. Data from "Inventory of Computers in U.S. Higher Education 1966-67" conducted by Southern Regional Education Board with NSF support. To be published by NSF, Fall, 1970.

The above programs were influenced by the "Preliminary" recommendation of the ACM Curriculum Committee but not by their final report "Curriculum 68". However, if we assume that there has been a linear increase in the number of degree programs, we get the following estimates for majors and graduates for June 30, 1969.

	Computer Science
# programs	49
# majors	3000
(1968-69) # graduates	750

Shortages of faculty candidates have no doubt slowed the development and initiation of many planned programs.

Because of one or two very well advertised bachelor's degree programs in small institutions, some have begun to believe that there are many such programs being attempted at such institutions. This is definitely not the case. In fact only 7 of the estimated 83 bachelor's programs (of all types) in operation as of June 30, 1967 were in institutions which did not offer at least a master's degree and had at least 10,000 students enrolled.

Of the 30 bachelor's degree programs reported in Computer Sciences going as of June 30, 1967, 23 different states are represented. Approximately two out of three of these programs are housed in a Computer Science Department or a joint department in its name (e.g. Information and Computer Science). In 1970 there are approximately 40 Computer Science Departments. Almost without exception these are located at well-known state and private universities.

Possible Trends in Bachelor's Programs in Computer Sciences. The ACM Curriculum Committee did not address itself to how the "Curriculum 68" recommendations could be used to structure a degree program for those who will work in the business community. This was deliberate. At the time it was felt that the needs of the "data processor" were incompatible with those of the "computer scientist" or at least that the intersection of the needs and interests was very small. It is quite possible that the recent developments which have led to much more complex hardware, and consequently software, systems have greatly expanded the intersection of interests. At least that was the feeling of some who participated in the conference whereas others maintained that all previous discussion (and the conference included) had not come to grips with the needs of the large majority of computer personnel who work in the business area.

To this end still another committee of ACM was appointed in 1965 to study and make recommendations for Computer Education for Information Processing Systems in Organizations. Dr. Dan Teichroew, University of Michigan is Chairman of this committee. A preliminary report is expected to be published in 1970. The activities of this committee is supported by an NSF grant to the Association for Computing Machinery.

A proposed undergraduate Computer Science program was presented at the conference by Dr. Alan Perlis of Carnegie-Mellon University, which looks promising and is one of the first attempts to satisfy this "need of the majority" within a Computer Science framework. The main "different" features of this proposed program are that three courses in Operations Research and a course in Administration and Finance are required.

Many of the conference participants felt that "Computer Science" degrees were desirable for those who manage or aspire to manage large business systems.

Although specific content and recommendations for a bachelor's degree program were not discussed it was the consensus of the group that considerable laboratory work should be included in a bachelor's program in Computer Science. Such laboratory experience should include work with a large data system.

Computer Science at the University of Waterloo * (J.W. Graham)

History and Philosophy. Computer Science courses have been taught at the University of Waterloo since the academic year 1959-60. They were actually taught before the university installed its first computer. It was not until 1964 that a program in Computer Science and a philosophy of operation of that program became evident in any formal sense.

At the present it is felt that although Computer Science is starting to get as a coherent body of knowledge it is still not at the stage where it can be taught as a complete undergraduate discipline and be considered as a basic body of knowledge such as mathematics. It is felt by the Faculty at Waterloo and by many others in the field [4] that

----

* Numbers in square brackets, e.g. [6] , refer to references found at the end of this subsection.

Computer Science should be taught as a specialized area in a basic discipline such as Mathematics or possibly Electrical Engineering. If Mathematics were chosen as the basic discipline a student would undertake a program such as the one recommended in the CUPM document [8] and then pursue some optional courses in his area of specialization whether it is algebra, geometry, statistics or computer science.

Computer Science courses were taught to undergraduate Engineering and Mathematics students at the University of Waterloo for the first time in the academic year 1960-61. A graduate course in programming had been given for the first time in 1959-60. These undergraduate courses were mostly applications-oriented and consisted primarily of numerical methods and analysis and some programming. The first course at the undergraduate level which was primarily programming was offered in 1961-62 to the graduate engineering class. These courses were taught by a group of three people who were in the mathematics department but who had a prime interest in Computer Science.

Computer Science course development remained at this level until the introduction of the Honours Co-operative Mathematics course in September 1964. (A co-operative course is established in such a way that students spend alternating four-month terms in school and working in industry and business. In this way they are able to gain practical industrial experience as well as a sound academic background. The academic content is identical to the regular mathematics programs already being offered at Waterloo.) This co-operative course had both Actuarial Science and Computer Science options and so a need arose to develop a more comprehensive curriculum in Computer Science to supplement the courses already offered. The courses at that time consisted of two numerical analysis and programming courses and courses in allied areas such as probability and statistics and logic. Any curriculum that was developed for the co-operative program was applicable to the regular program and vice-versa.

The honours mathematics program with Computer Science option has now been established on both a regular and co-operative basis since 1964, although it is still under development and will continue to change in order to remain current. The present program is based on a solid foundation of mathematics with the optional courses primarily in later years. There are three ways a student may pursue an option in Computer Science. He may enroll in the Co-operative Honours Mathematics Program, the Regular Honours Mathematics Program or the Regular General Mathematics Program. In an honours program a student attends university for four years and completes 17 or 18 mathematics courses (including Computer Science) and 9 elective courses. The difference between the regular and co-operative program has been explained previously. In a general program he attends university for three years and completes 9 mathematics courses and 7 elective courses. The elective courses may be chosen from the sciences, humanities or engineering and the student may study such courses as philosophy, psychology, economics, chemistry, physics, etc.

A description of the undergraduate program on a year-by-year basis is presented next. Where a choice of Mathematics courses is indicated some typical examples are shown; elective courses are indicated where taken, but no course titles will be given.

Regular and Co-operative Honours Program in Mathematics for Students Specializing in Computer Science

First Year

Mathematics 130	Calculus
Mathematics 131	Algebra and Solid Geometry
Mathematics 132	Introduction to Computer Science

and 3 elective courses

Second Year

Mathematics 229	Linear Algebra
Mathematics 233	Probability and Statistics
Mathematics 237	Differential and Integral Calculus
Mathematics 240	Applications in Computer Science

and 3 Elective courses one of which may be another Mathematics course such as Mathematics 234 (Mechanics) or Mathematics 235 (Actuarial Mathematics). The co-operative program students must select 235.

Third Year

Five Mathematics courses including:

Mathematics 329	Abstract Algebra
Mathematics 332	Theory of Functions

and at least one of

Mathematics 334	Numerical Analysis
Mathematics 340	Computer Systems

Students in the Co-operative program must take 334 and 340, and students in the regular course usually choose both of these courses.

The remaining Mathematics courses to be selected might be chosen from:

Mathematics 333	Differential Equations
Mathematics 338	Mathematical Statistics
Mathematics 351	Combinatorial Mathematics
Mathematics 352	Mathematical Operations Research

2 Elective courses.

<u>Fourth Year</u>

Five Mathematics courses.

Some typical examples are:

Mathematics 471a	Switching Circuits
Mathematics 471b	Computer System Organization and Logic Design
Mathematics 472a	Introduction to Automata Theory
Mathematics 470	Numerical Solution of Ordinary and Partial Differential Equations
Mathematics 457	Applied Combinatorial Mathematics
Mathematics 436	Mathematical Logic
Mathematics 455	Mathematical Programming

and 2 Elective courses

<u>Regular General Program in Mathematics for Students Specializing in Computer Science</u>

<u>First Year</u>

The first year is common to both honours and general programs.

Second Year

Mathematics 229	Linear Algebra
Mathematics 237	Differential and Integral Calculus
Mathematics 240	Applications in Computer Science

and 2 Elective courses

Third Year

Three Mathematics courses including at least one of:

Mathematics 334	Numerical Analysis
Mathematics 340	Computer Systems

Other courses may be selected from the list in the Honours program.

2 Elective courses.

The previous material describes the programs in Computer Science at the University of Waterloo. More comprehensive course descriptions are given later.

The course as described in this paper is the one currently being taught. It is not considered complete and there is still development to be done apart from the continual changing of courses to keep them current. The first two years of the program will probably remain essentially unchanged for the present time; the third year will undergo moderate changes as ideas in curriculum become stable; the fourth year requires some additional courses before it can be considered completely viable. For instance, one or two courses in the area of programming and programming languages will be necessary before the final year of the program is considered to be complete.

Undergraduate Course Descriptions at the University of Waterloo

132. Introduction to Computer Science. A thorough introduction to algorithms, stored-program computers and programming languages. Concept and properties of an algorithm, language and notation for describing algorithms. Analysis of computational problems and development of algorithms for their solution. A procedure-oriented language (FORTRAN IV) and machine and assembly languages are used to implement algorithms on the computer.

2 hours lectures, 2 hours problems.

240. Applications in Computer Science. Some numerical methods are introduced in the first term and programmed for the computer using FORTRAN IV. Concepts of Numerical Errors. Methods in Interpolation, numerical integration, solution of nonlinear equations, linear systems of equations.

In the second term non-numeric computing is introduced including simulation, the concept of a list and elementary list processing techniques, sorting, symbol manipulation.

2 hours lectures, 1 hour problems.

334. Numerical Methods. Finite differences. Solution of equations. Computation with series and integrals. Linear systems and matric methods. Difference equations and relaxation methods. Numerical solution of partial differential equations. Methods employing electronic computers.

2 hours lectures, 2 hours laboratory.

340. Computer Systems. A discussion of the hardware and software required in a computer system. Overlapped channels, interrrupt facilities, memory protection, buffers, input-output control systems, macro-programming, monitor systems. Relocation schemes, multi-programming, multi-processing, dynamic memory allocation, time-sharing. Special purpose computer systems and simulation of computer systems. Peripheral equipment. Introduction to the theory of sequential machines and the logical design of computers.

2 hours lectures.

471b. Computer System Organization and Logic Design.
Part I. Logic Design: Brief review of basic switching theory; number systems; logic circuits; storage elements, standard computer subsystems; overall logical design of a conventional processor; automated design of processors - computer compilers.
Part II. System Organization: Definition of total system requirements for batch processing, time-sharing, real-time processing and other applications; hardware-software tradeoffs and the impact of large-scale integration; unorthodox hardware organizations; evaluation of solutions. Some attention will be paid to design for reliability and fault diagnosis as additional system requirements.
Prerequisite: Switching Circuits. (Math 471a)

2 hours lectures

472a. Introduction to Automata Theory. Sequential machine models. Reduction of completely and incompletely specified machines. Decomposition of machines. Nondeterministic machines. Finite automata and regular events. Multitape finite-state nonwriting automata.

Prerequisite: Consent of instructor.

2 hours lectures, Fall Term.

472b. Introduction to Turing Machines and Computability Theory. Formalisms for Turing machines. Variants of Turing machines. Universal Turing machines. Unsolvability of the halting problem and related results. Recursive and primitive recursive functions. Models similar to digital computers. Brief introduction to formal languages. Post's symbol-manipulation systems. Simple bases for computability.

Prerequisite: Mathematics 472a or Consent of instructor.

2 hours lectures, Winter Term.

Graduate Course Descriptions at the University of Waterloo

Mathematics 510

Survey of Numerical Analysis

Interpolation and approximation. Quadrature. Solution of initial-value and boundary-value problems. Solutions of linear and non-linear algebraic systems.

Mathematics 511

Introduction to Computer Programming

Introduction to algorithm implementation. Basic concepts of computers and computer programming using the SPECTRE machine as a basis. Programming in a Problem-Oriented Language FORTRAN IV. Assembler and macro programming using /360 Assembler or equivalent.

Mathematics 870

Numerical Solution of O.D.E.

Initial-value problems. Existence and uniqueness. One-step methods. Error analysis. Multi-step methods. Boundary-value problems.

Mathematics 871

Numerical Methods in Linear Algebra

Direct methods of solving linear equations and inverting matrices. General discussion of iterative methods for linear equations and some iterative methods for inverting matrices. The problems of ill-conditioned matrices and the discussion of errors. The iterative and direct methods for finding eigenvalues. Problems of identical and close eigenvalues.

Mathematics 872

Numerical Solution of Partial Differential Equations

Classification of Equations. Characteristics. Discrete-variable methods for Parabolic and Hyperbolic P.D.E. Systems of P.D.E. - Reduction to canonical form and numerical solutions.

Mathematics 885

Time-sharing Systems

A two-term course on problems and techniques of large scale time-sharing systems. Topics covered include: evolution of computer systems; description of some early systems; overview. Fundamental concepts: resources; virtual memory; virtual machines; time-slicing; system availability. Fundamental problems: resource management; resource sharing; tradeoffs; storage allocation; device allocation; path finding; time allocation; system metering. System facilities: communications facilities; file management facilities; program production facilities - languages, language processor control, dynamic loading and binding, debugging facilities. System hardware architecture. System construction; scaffolding; interrupt logging. Case studies of major time-sharing attempts: TTS/360, MULTICS, BASIC, APL/360 and the SP-1 Electronic Switching System.

Mathematics 886A

Computer Systems I

A discussion of the hardware and software required in a computer system. Including overlapped channels, interrupt facilities, memory protection, buffers, input-output control systems, macro-programming, multi-processing, dynamic memory allocation, time-sharing.

Mathematics 886B

Computer Systems II

Part 1 - Logic Design: Brief review of basic switching theory; number systems, logic circuits; storage elements, standard computer subsystems; overall logical design of a conventional processor; automated design of processors - computer compilers.

Part 2 - System Organization: Definition of total system requirements for batch processing, time-sharing, real-time processing and other applications; hardware-software tradeoffs and the impact of large-scale integration; unorthodox hardware organizations; evaluation of solutions. Some attention will be paid to design for reliability and fault diagnosis as additional system requirements. The faculty's APL/360 time-sharing facility will be used for demonstrations, problems, and term paper studies.

Mathematics 890

Programming Languages

This course will cover basic programming concepts as found in a number of high level programming languages. The student will be expected to write programs in many of the languages covered. The basic prerequisite is a good knowledge of programming in at least one programming language with course M892A a definite asset. The course content will involve the following high level languages: ALGOL, CPL, SNOBOL, LISP, PL1, SIMULA, GPSS.

Mathematics 891A

Design of Translators

Some theoretical aspects of mechanical language translation with references to compilers and compiling techniques both past and present, including general compiler organization, statement decomposition, syntax analysis, code optimization and compiler compilers.

Mathematics 892A

Data Structures I - Basic Data Structures with Applications

Data structures from simple variables and arrays to lists, trees, rings and plexes. Applications to programming languages, computer graphics and data management systems.

Mathematics 892B

Data Structures II

Programming techniques for creation, deletion (garbage collection) and manipulation of different types of lists and trees. There application to symbol manipulation. An introduction to the uses of PL/1, Formula Algol and other Languages will be given.

Mathematics 893

Data Structures III

Mathematics 894A

Automata Theory

Reduction of completely and incompletely specified machines. Decomposition of machines. Nondeterministic machines. Finite automata and regular events. Multitape finite-state nonwriting automata.

Mathematics 894B

Automata Theory

Regular expressions and languages. Basic properties, derivative techniques, characterization of finite automata and sequential machines by regular expressions. Sequential circuits: analysis and synthesis techniques. Special classes of finite automata and languages (definite, linear, star-free etc.). Algebraic properties of regular languages. Further generalizations of finite automata.

Mathematics 897A

Formal Languages I

Phrase-structure grammars and languages. The Chomsky hierarchy of languages. Context-free languages: basic properties, normal forms, ambiguity. Special types of context-free languages. Finite-state languages and finite automata. Turing machines and type 0 languages. Deterministic and non-deterministic pushdown automata and corresponding languages. Context-sensitive grammars and linear bounded automata. Text: Hopcroft and Ullman, "Formal Languages and their Relation to Automata", Addison-Wesley, 1969.

Mathematics 897B

Formal Languages II

Operations on languages. Closure properties. Abstract families of languages. Solvable and unsolvable problems for type 0, context-sensitive, context-free and regular languages. Syntactical analysis of context-free languages. Precedence grammars, LR(k) and LL(k) grammars. Deterministic syntactical analysis. Syntax directed translations. Generalized context-free grammars. Formalization of language semantics.

Mathematics 898

Computability Theory

The notions of effective computability and computational complexity. Turing machines, their formalisms and variants. Universal Turing machines and other bases for computability. Machine-based complexity studies; time and space bounds. Machine organization vs. computation time. Primitive recursive functions and their characterization as loop programs. Church's thesis. Basic unsolvability results. Recursively enumerable sets; productive and creative sets. Relative computability. The recursion theorem. The speed-up and gap theorems. Priority arguments; the Friedberg-Muchnik theorem. The arithmetic hierarchy.
Reference: H. Rogers, J., Theory of Recursive Functions and Effective Computability, McGraw-Hill, 1968.

Mathematics 902A

Artificial Intelligence I

Some topics from mathematical logic: instances (λ-calculus), proof procedures. Formal languages for mechanical theorem proving. Complete methods for theorem proving: Goutzen type methods, Herbrand type methods; resolution methods. Some heuristic approaches.

Mathematics 902B

Artificial Intelligence II

Problems of analysis of quasi natural languages and question answering systems. Methods of problem solving. Mechanical game playing. Learning processes in problem solving and game playing.

Mathematics 905B

Seminar on Automata and Language Theory

References

1. Caffrey, J., Mosmann, C. T., Computers on Campus, American Council on Education, Washington, D. C., 1967.

2. Cress, P. H., Dirksen, P. H., Graham, J. W., FORTRAN IV with WATFOR, Applied Analysis and Computer Science Department, University of Waterloo, Prentice-Hall, 1968.

3. Cress, P. H., Dirksen, P. H., et al., Description of /360 WATFOR, A FORTRAN IV Compiler, Technical Report, CSTR-1000. Applied Analysis and Computer Science Department, University of Waterloo.

4. Curriculum 68 - Recommendations for Academic Programs in Computer Science - A Report of the ACM Curriculum Committee on Computer Science, CACM, March 1968, pp. 151.

5. IBM 1620 GOTRAN Interpretive Programming System, Form No. C26-5594-0, IBM 1961.

6. Ledley, R. S., FORTRAN IV Programming, McGraw-Hill, 1965.

7. McCracken, D. D., A Guide to FORTRAN Programming, John Wiley, 1961.

8. Recommendations on the Undergraduate Mathematics Program for Work in Computing, Committee on the Undergraduate Program in Mathematics, May 1964.

9. Shantz, P. W., German, R. A., Mitchell, J. G., Shirley, R. S. K., and Zarnke, C. R., WATFOR - The University of Waterloo FORTRAN IV Compiler, CACM, January 1967, pp. 41.

10. Shantz, P. W., Tutor. Applied Analysis and Computer Science Department, University of Waterloo.

A Proposed Undergraduate Computer Science Program at Carnegie-Mellon University

(A.J. Perlis)

		Freshman	Sophomore	Junior	Senior
1st Sem.	1.	Anal I	Alg I	Prob & Stat I	OR II
	2.	Prog I	Prog III	Comp.Sys. II	Abstr.Sys. III
	3.	Phys I	Anal III	Lab II	Elect. II
	4.	Hum.	OR I	Abstr.Sys. I	Elect. III
	5.	Hum.	Hum.	Hum.	Hum.
2nd Sem	1.	Anal II	Lab I	Prob & Stat II	OR III
	2.	Prog II	Alg II	Abstr Sys. II	Comb. Anal.
	3.	Phys II	Prog IV	Lab III	Administration and finance
	4.	Hum.	Comp. Sys. I	Elect. I	Elect. IV
	5.	Hum.	Hum.	Hum.	Hum.

NOTES:

Hum	= Humanities
Prob & Stat	= Probability and Statistics

Programming I -- IV

1. Algorithms, programs and language (organized by data
2. Algorithms, programs and language structures)
3. Machines and their programs
4. Problems associated with the management of programs: file systems, libraries; and proofs of termination and correctness; Verification, representation and documentation of programs.

Computer Systems I and II

1. Devices
2. Representation
3. Synthesis
4. System design

Abstract Systems I to III

Logic: Propositional Calculus; 1st order Predicate Calculus
Automata Theory: Finite state machines and regular expressions
Turing machines
Computability
Stages of Computability
Math, Linguistics, correspondences (recognizers as machines)

Operations Research

OR I Optimization Techniques
OR II Simulation Techniques and modeling
ORIII Processing requirements of large data systems

Computer Science Laboratory I - III

1. Building, enhancing, auditing a sub-routine library
2. Interfacing two systems
3. Design of a system
4. Completion of a system
5. Managing a system design and construction

A Professional Master's Program at Stanford University (E.J. McCluskey)

In response to the demand for a professional degree for students interested in the design of hardware-software computer systems, a special degree program has been devised. Students may enroll in either the Computer Science or Electrical Engineering Department. For Computer Science students the degree obtained bears the designation Master of Science in Computer Science: Computer Engineering. In Electrical Engineering, the degree designation is Master of Science in Electrical Engineering: Computer Engineering. Students should indicate a preference for this degree when applying for admission.

A program in Computer Engineering should include 42 units of work, of which at least 36 must be graded. These will normally come from the following courses: CS 135 Numerical Methods (or both CS 137 and 138 Numerical Analysis), CS 109 Assembly Language Programming, CS 111 (EE 181) Introduction to Computer Organization, CS 112 (EE 182) Digital Computer Organization (or both EE 281 Theory of Switching and EE 282 Logic Design), CS 140A, B (EE 286A,B) Systems Programming, CS 144A Data Structures, CS 246 (EE 386) Operating Systems, CS 206 Computing with Symbolic Expressions, CS 150 Introduction to Combinatorial Theory (or CS 155 Concrete Mathematics, or some course in discrete mathematics), OR 252 Operations Research, CS 298 Software Engineering Laboratory (or 6 units of CS 293 Computer Laboratory or 6 units of EE 390 Special Studies), and EE 380 Seminar on Digital System.

This program is open to students with a scientific bachelor's degree (a BS in Mathematics, Statistics, Physics, or Engineering); or with a degree having a mathematical background (courses in calculus, a knowledge of linear algebra, and probability). Some knowledge of programming will be required.

Students requiring remedial help for an inadequate background in programming should enroll in the basic programming course, CS 106, during the Summer Quarter preceding entrance into this program. Math 113 Linear Algebra and Matrix Theory, and Stat 116 Probability Theory, or their equivalents, may be taken while the student is a candidate; however, credits for these courses will not count towards the units necessary for this degree.

The Computer Engineering program will begin in Autumn Quarter each year to enable a full-time student to complete the degree in one academic year. Honors Cooperative students should be able to complete the program in two normal academic years plus one Summer Quarter.

The degree in Computer Engineering is intended as a terminal degree. Students planning to obtain the PhD degree are advised to apply directly for admission to the PhD program in either the Computer Science Department or the Electrical Engineering Department.

<u>Computer Systems Laboratories</u> (W.F. Miller, C.L. Coates, R. Andree, F. Gruenberger, R. Spinrad, A. Forsythe, S. Seely)

In laboratory courses, students are expected to work in teams of about six students under close supervision of a faculty member and a teaching assistant. The student team is expected to concentrate on design, documentation, scheduling of their work, performance evaluation, efficiency, error recovery, diagnostics, maintainability and other features of a well-engineered system. It is expected that each student should take the equivalent of two of the laboratories described below during the course of this study.

We propose the following computer systems laboratory courses as basic to a graduate Computer Science departmental offerings:

CS Lab. 1. Construction of Assemblers and Compilers

CS Lab. 2. Construction of Operating Systems

CS Lab. 3. Construction of Terminal Systems (both typewriter and graphics)

CS Lab. 4. Construction of Switching, Communication and Process Control

CS Lab. 5. Construction of Large Data Base Systems

In addition, we consider two additional laboratory courses that could be given to or in place of the above five:

CS Lab. 6. Management of a Computer Facility

CS Lab. 7. Construction of Large Application Systems

The above laboratory courses, particularly the first five, are graduate-level courses given concurrently with or following a lecture course covering the subject matter. It is intended that the lecture course cover the theory, models, and formal aspects of the subject matter. The associated laboratory is intended to provide the student an experience that will sharpen his understanding of the theory and will give him an understanding of the practical problems of implementing large systems.

The companion lecture courses associated with the above listed laboratory courses are given below:

Laboratory Course	Lecture
C.S. Lab. 1. Construction of Assemblers and Compilers	Lecture course such as I5 and/or A1 from Curriculum 68, A Report of the ACM Curriculum Committee on Computer Science. Includes definition of formal grammars, arithmetic expressions and precedence grammar, algorithms for syntactic analysis, recognizers, semantics of grammar, object code generation, organization of assemblers and compilers, meta-languages and systems.
C.S. Lab. 2. Construction of Operating Systems	Lecture course such as I4 and/or A2 and/or A3 from Curriculum 68. Includes operating systems characteristics, structure of multiprogramming systems, structure of time-sharing systems, addressing structures, interrupt handling, resource management, scheduling, file system design and management, input-output techniques, design of system modules, sub-systems.

C.S. Lab. 3. Construction of Terminal Systems (both type-writer and graphics)	Lecture course such as I4 and A6. Includes text editors, string manipulations, data structures for text editors, job control languages, data structure for pictures, syntax and semantics of terminal and graphics language, control of the console system, meta-language and systems.
C.S. Lab. 4. Construction of Switching, Communication Systems, and Process Control	Lecture course such as I4 and/or A2 of Curriculum 68. Includes traffic control, interprocess communication, system interfaces, realtime data acquisition, asynchronous and synchronous control, telecommunication, analog-to-digital and digital-to-analog conversion.
C.S. Lab. 5. Construction of Large Data Base Systems	Lecture course such as A5 and A8 of Curriculum 68. Includes organization of large data base systems, data organization and storage structure techniques, data structuring and inquiry languages, searching and matching, automatic retrieval, dictionary systems, question answering.

These laboratories will require a certain amount of "hands on" use of a substantial computer facility. In some installations it may be possible to carry out the entire project in a subsystem or partition of a larger system. In that case the use of the subsystem would have to be dedicated to the project for a substantial portion of time.

These laboratories are presented as examples of laboratories that might be given. Each school will have different staff and facilities available and will present variations on this proposal. The important emphasis is the supervised hands-on experience with attention to the practical aspects of the system.

Retreading (G.E. Forsythe)

There is a major national shortage of people educated in Computer Science. To begin to meet this crisis will require the yearly addition of 3,500 persons to the level of a master's in Computer Science and 500 at the level of a PhD in Computer Science. The current educational capabilities do not permit this need to be met. Each year U.S. universities are graduating some 1100 persons with a PhD in physics. Many of these persons have the talent and desire to become faculty members at good universities, or research staff members at good research laboratories. For a variety of reasons there is insufficient demand at present for these persons in their own field. Many physicists are interested and able to turn their talents to a career in computing. We believe that many of the recent PhD's in physics can be converted into good Computer Scientists in about two years, and into Computer Science faculty members in about three years. In many cases these times seem significantly less than the corresponding times required for students entering graduate school in Computer Science. Moreover, the annual amount of faculty time required for these post-doctoral fellows would appear to be about half to two-thirds of that required for supervising graduate students.

The principal advantage of this approach is the speed-up in creating new computing experts as compared to one starting with conventional new graduate work. However, one cost would be the substantially larger salaries required for post-doctoral students than for regular graduate students. Furthermore, if there were an overload of, say, 100 post-doctoral students, there would be a substantial cost for faculty members to deal with them.

What has been said about physicists may apply also to mathematicians and, with lesser force, to some other fields.

Computer Science and Related Degree Programs in U.S. Higher Education (J.W. Hamblen)

The number of institutions reporting various degree programs, their total faculty, numbers of majors, both undergraduate and graduate, and the number of degrees awarded during 1966-67 are reported in Table 1. These numbers were obtained in an inventory conducted by the Southern Regional Education Board (SREB). Best estimates for each heading are also given. These estimates were obtained by applying the overall extrapolation ratio of 1.25 to the reported totals. This is probably a little high for the higher degree levels but the estimates are likely to be within 10% of the true values on the high side.

In 1964-65 the institutions of higher education projected that they would have 18,807 undergraduate majors and 5318 graduate majors during the academic year 1968-69. However, the estimates from Table 1 show that by 1967-68 the undergraduate figures had already been exceeded (22,161) and that the number of graduate majors (4936) was fast approaching the 1968-69 projections. These comparisons are summarized in Table 2. along with data gathered by the Southern Regional Education Board (SREB) on the number of students actually enrolled in 1964-65.

Table 3 provides a comparison with the population estimates of going programs during 1964-65, those projected by the institutions for about 1967-68, and the 1966-67 estimates. This table shows that except for the associate degree programs, which had already in 1966-67 exceeded the projected figure (188) for 1967-68, the numbers of new degree programs were lagging behind the numbers projected by the institutions. It follows then that the programs in existence during 1966-67 were accomodating many more majors than was originally anticipated by the institutions when they made their projections for 1968-69.

Table 4 lists the schools offering degree programs in Computer Science, data processing,information science, information systems and information processing.

Table 5 is a separate listing of schools offering a bachelor's degree in Computer Science, and those offering PhD degrees in Computer Science, data processing, information science and information systems.

Figure 1 originally appeared in an article published in the ACM COMMUNICATIONS, April 1964, entitled "Status of Computer Sciences Curricula in Colleges and Universities". It represents an attempt to show relationships among the various computer related program names and some other more established academic areas. It is to be considered a point of departure and not necessarily an end.

Table 6 lists the schools offering associate degree programs in data processing, Computer Science, and their related areas.

Table 1
Computer Oriented Degree Programs
1966-67

	Number of Insts.*	Faculty	1966-67 Majors Under-Grads.	Grads.	1966-67 Graduates Assoc.	Bach.	Mast.	Doct.
Data Processing	133	560	12,765	92	872	49	13	--
Computer Science	059	569	1,727	1,429	58	175	242	34
Option in Elec. Eng.	012	108	447	298	--	127	86	8
Info. Science	010	122	100	627	--	--	64	3
Option in Math.	008	91	863	780	--	12	5	73
Computer Technology	008	36	496	--	13	20	--	--
Option in Eng.	007	53	35	268	4	--	9	3
Computer Programming	004	6	281	--	78	--	--	--
Information Systems	004	84	424	8	--	19	3	--
Management Science	003	43	--	125	--	--	91	15
Option in Ind. Eng.	003	31	115	26	--	41	13	--
Option in Bus. Adm.	002	14	50	225	--	25	50	1
Systems Analysis	002	7	313	--	--	--	--	--
Information Processing	001	5	--	23	--	--	3	--
Quantitative Methods	001	3	--	--	--	--	--	--
Systems Engineering	001	11	113	48	--	7	15	2
TOTAL	258	1,743	17,729	3,949	1,025	475	539	139
Estimated Population Totals	311	2,179	22,161	4,936	1,281	594	674	174

* Institutions which offered a particular degree program at more than one level are counted only once.

Table 2

Estimated Numbers of Majors in Computer Sciences
Data Processing, Information Sciences, etc.

Source	Year	# Majors Undergraduate	Graduate
SREB Survey	1964-65	4,338	1,314
SREB Survey	Projected 1968-69	18,807	5,318
SREB Inventory (Table 1)	1966-67	22,161	4,936

Table 3

Estimates of Numbers of Degree Programs in
Computer Science, Data Processing, Information Science, etc.

Source	Year	Status	Assoc.	Bach.	Mast.	Doct.	Total
SREB Survey	1964-65	Going	83	44	61	38	226
SREB Survey	1964-65	New, Planned by 1967-68	105	107	76	43	331
SREB Survey	1967-68	To Be Going	188	151	137	81	557
SREB Inventory	1966-67	Going	192	83	87	52	414

TABLE 4
SOUTHERN REGIONAL EDUCATION BOARD
COMPUTER SCIENCES PROJECT
NSF INVENTORY OF COMPUTERS

Degree Programs Offered by Listed Universities in Computer Science:

SCHOOL	LOCATION	DEGREES OFFERED
Auburn University	Auburn, Alabama	B.
University of Arkansas	Fayetteville, Arkansas	M.
Stanford University	Palo Alto, California	M., D.
University of California	Berkeley, California	B., M., D.
University of California	Santa Cruz, California	B.
U. S. Air Force Academy	Colorado Springs, Colorado	B.
Florida Inst. of Technology	Melbourne, Florida	B.
Georgia State College	Atlanta, Georgia	M.
Bradley University	Peoria, Illinois	M.
Northwestern University	Evanston, Illinois	M., D.
University of Illinois	Urbana, Illinois	B., M., D.
Purdue University	Lafayette, Indiana	B., M., D.
Iowa State University	Ames, Iowa	B., M., D.
University of Iowa	Iowa City, Iowa	M., D.
Kansas State University	Manhattan, Kansas	B., M.
University of Kentucky	Lexington, Kentucky	B.
Louisiana Poly. Institute	Ruston, Louisiana	B.
University of Maryland	College Park, Maryland	M.
University of Massachusetts	Amherst, Massachusetts	M.
Michigan State University	East Lansing, Michigan	B.
University of Minnesota	Minneapolis-St. Paul, Minnesota	M., D.
University of Southern Mississippi	Hattiesburg, Mississippi	B.
Univ. of Missouri at Rolla	Rolla, Missouri	B., M., D.
Washington University	St. Louis, Missouri	M., D.
Princeton University	Princeton, New Jersey	M., D.
Rutgers State University	New Brunswick, New Jersey	M.
Stevens Inst. of Technology	Hoboken, New Jersey	M.
New Mexico Highlands Univ.	Las Vegas, New Mexico	B., M.
N. M. Inst. of Mining Technology	Socorro, New Mexico	B.
New Mexico State University	University Park, New Mexico	M.
Cornell University	Ithaca, New York	M., D.
Rensselaer Polytechnic Institute	Troy, New York	M., D.
State University of New York	Albany, New York	M.
State University of New York	Brooklyn, New York	B.
State University of New York	Potsdam, New York	B.
Union College	Schenectady, New York	B., M.
North Dakota State University	Fargo, North Dakota	B.
University of Dayton	Dayton, Ohio	B.
Youngstown State University	Youngstown, Ohio	B.
Oregon State University	Corvallis, Oregon	B., M., D.
Carnegie-Mellon University	Pittsburgh, Pennsylvania	D.
Pennsylvania State University	University Park, Pennsylvania	B., M., D.
University of Pittsburgh	Pittsburgh, Pennsylvania	M.
Brown University	Providence, Rhode Island	M., D.

TABLE 4

Degree Programs in Computer Science (cont'd):

University of South Carolina	Columbia, South Carolina	B., M.
Winthrop College	Rock Hill, South Carolina	B.
Middle Tennessee State Univ.	Murfreesboro, Tennessee	B.
Texas A & M University	College Station, Texas	M.
University of Houston	Houston, Texas	B., M.
University of Texas	Austin, Texas	M., D.
University of Utah	Salt Lake City, Utah	B., M., D.
Utah State University	Logan, Utah	B.
University of Virginia	Charlottesville, Virginia	M., D.
University of Washington	Seattle, Washington	M., D.
University of Wisconsin	Madison, Wisconsin	B., M., D.

Degree Programs Offered by Listed Universities in Data Processing:

California State Poly. College	Pomona, California	B.
Kansas State College	Pittsburg, Kansas	B.
Kansas State Teacher's College	Emporia, Kansas	B., M.
Louisiana Polytechnic Institute	Ruston, Louisiana	B., M.
Ferris State College	Big Rapids, Michigan	B.
Mississippi State University	State College, Mississippi	B., M., D.
Eastern New Mexico University	Portales, New Mexico	B.
Pace College	New York, New York	B.
Rensselaer Polytechnic Institute	E. Windsor Hill, New York	M.
Northern Oklahoma College	Tonkawa, Oklahoma	B.
University of Chattanooga	Chattanooga, Tennessee	B.
West Texas State University	Canyon, Texas	B.

Degree Programs Offered by Listed Universities in Information Science:

Georgia Inst. of Technology	Atlanta, Georgia	M.
University of Chicago	Chicago, Illinois	M.
Syracuse University	Syracuse, New York	M., D.
University of North Carolina	Chapel Hill, North Carolina	M., D.
Ohio State University	Columbus, Ohio	B., M., D.
University of Dayton	Dayton, Ohio	M.
Lehigh University	Bethlehem, Pennsylvania	D.
Point Park College	Pittsburgh, Pennsylvania	B.
University of Pennsylvania	Philadelphia, Pennsylvania	M., D.
Washington State University	Pullman, Washington	M., D.

Degree Programs Offered by Listed Universities in Information Systems:

California State College	Los Angeles, California	B.
University of Maryland	College Park, Maryland	B.
Northeastern University	Boston, Massachusetts	B.
Lehigh University	Bethlehem, Pennsylvania	M.

Degree Programs Offered by Listed Universities in Information Processing:

Southern Illinois University	Carbondale, Illinois	M.

TABLE 5
SOUTHERN REGIONAL EDUCATION BOARD
COMPUTER SCIENCES PROJECT
NSF INVENTORY OF COMPUTERS

BACHELOR'S DEGREE PROGRAMS IN COMPUTER SCIENCE:

Auburn University
University of California at Berkeley
University of California at Santa Cruz
Air Force Academy
University of Illinois
Purdue University
Iowa State University
Kansas State University
University of Kentucky
Louisiana Polytechnic Institute
Michigan State University
University of Southern Mississippi
University of Missouri at Rolla
New Mexico Highlands University
New Mexico Institute of Mining Technology
State University of New York Downstate Medical Center
State University of New York College--Potsdam
Union College
North Dakota State University
University of Dayton
Youngstown State University
Oregon State University
Penn State University
University of South Carolina
Winthrop College
Middle Tennessee State University
University of Houston
Utah State University
University of Utah
University of Wisconsin

Ph.D. PROGRAMS:

DATA PROCESSING:

Mississippi State University, State College, Mississippi, 39762

COMPUTER SCIENCE:

Stanford University, Palo Alto, California, 94305
University of California at Berkeley, Berkeley, California, 94720
Northwestern University, Evanston, Illinois, 60201
University of Illinois, Main Campus, Urbana, Illinois, 61801
Purdue University, Lafayette, Indiana, 47907
Iowa State University of Science Technology, Ames, Iowa, 50010
University of Iowa, Iowa City, Iowa, 52240
University of Minnesota, Minneapolis-St. Paul, Minnesota, 55455

TABLE 5

Ph.D. PROGRAMS IN COMPUTER SCIENCE (cont'd):

University of Missouri at Rolla, Rolla, Missouri, 65401
Washington University, St. Louis, Missouri, 63130
Princeton University, Princeton, New Jersey, 08540
Cornell University, Main Campus, Ithaca, New York, 14850
Rensselaer Polytechnic Institute, Main Campus, Troy, New York, 12181
Oregon State University, Corvallis, Oregon, 97331
Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15213
Pennsylvania State University, University Park, Pennsylvania, 16802
Brown University, Providence, Rhode Island, 02912
University of Texas at Austin, Austin, Texas, 78712
University of Utah, Main Campus, Salt Lake City, Utah, 84112
University of Virginia, Charlottesville, Virginia, 22903
University of Washington, Seattle, Washington, 98105
University of Wisconsin at Madison, Madison, Wisconsin, 53706

INFORMATION SCIENCE:

Georgia Institute of Technology, Atlanta, Georgia, 30332
University of Chicago, Chicago, Illinois, 60637
Syracuse University, Syracuse, New York, 13210
University of North Carolina, Chapel Hill, North Carolina, 27514
Ohio State University, Columbus, Ohio, 43210
Lehigh University, Bethlehem, Pennsylvania, 18015
University of Pennsylvania, Philadelphia, Pennsylvania, 19104
Washington State University, Pullman, Washington, 99163

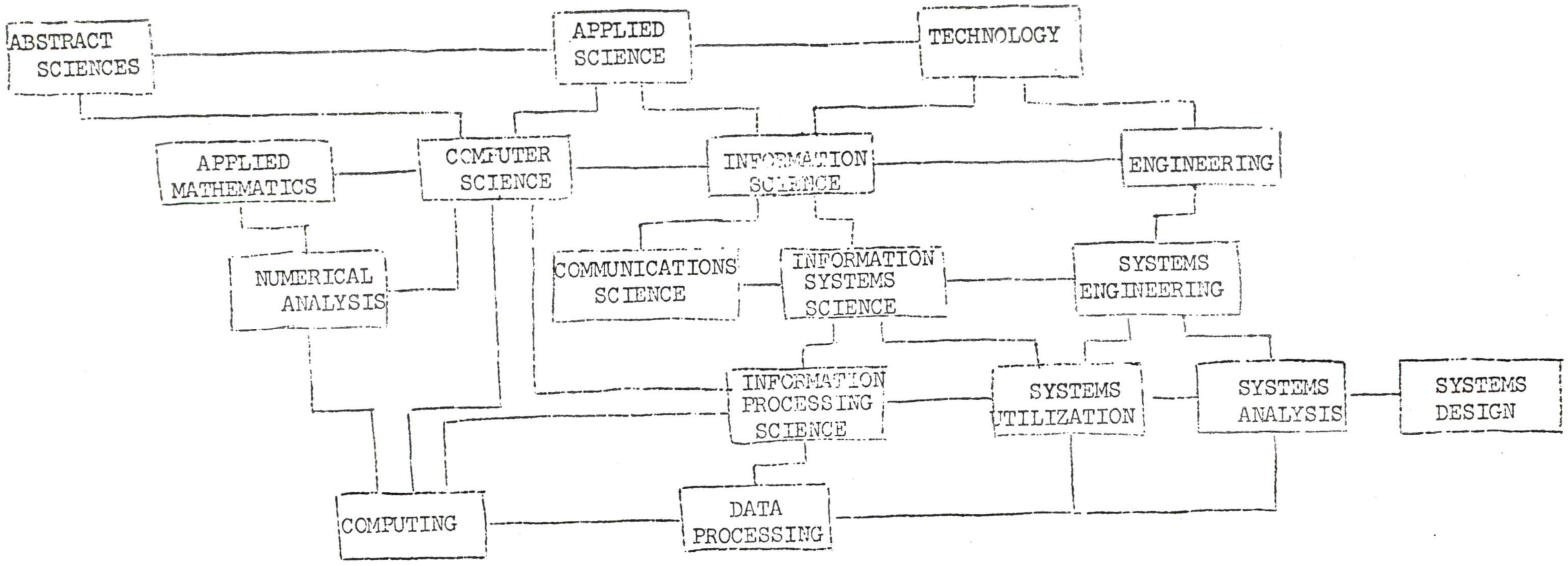
INFORMATION SYSTEMS:

Lehigh University, Bethlehem, Pennsylvania, 18015

Recent Additions:

University of Maryland, College Park, Maryland, 20742
University of Oklahoma, Norman, Oklahoma, 73069

FIGURE 1



HIERARCHY OF COMPUTER-RELATED FIELDS

TABLE 6

Colleges Offering an Associate Degree
in
Data Processing

SCHOOL	LOCATION	ZIP CODE
John C Calhoun State Tech College	Decatur, Alabama	35699
Wm L. Yancey State Jr. College	Bay Minette, Alabama	36507
Arizona Western College	Yuma, Arizona	85364
Phoenix College	Phoenix, Arizona	85013
Allan Hancock College	Santa Maria, California	93454
Bakersfield College	Bakersfield, California	93305
Cerritos College	Norwalk California	90650
Chabot College	Hayward, California	94545
Chaffey College	Altaloma, California	91701
San Mateo Jr. College Dist	San Mateo, California	94402
Contra Costa College	San Pablo, California	94806
Diablo Valley College	Pleasant Hill, California	94523
El Camino College	El Camino College, California	90506
Foothill Jr College Dist	Los Altos Hills, California	94022
Los Angeles Tr Tech College	Los Angeles California	90015
Orange Coast College	Costa Mesa, California	92699
Pasadena City College	Pasadena, California	91106
San Diego Junior College	San Diego California	92101
San Jose City College	San Jose, California	95114
Southwestern College	Chula Vista, California	92010
Mesa College Main Campus	Grand Junction Colorado	81501
Otero Jr College	La Junta, Colorado	81050
Southern Colorado St. College	Pueblo, Colorado	81005
Trinidad State Jr. College	Trinidad Colorado	81082
Norwalk Community College	Norwalk Connecticut	06854
Norwalk State Tech. Institute	S. Norwalk, Connecticut	06854
Thames Valley St. Tech College	Norwich, Connecticut	06360
Jr College of Broward County	Fort Lauderdale, Florida	33314
Miami-Dade Junior College	Miami, Florida	33156
North Florida Jr College	Madison, Florida	32340
Pensacola Jr College	Pensacola, Florida	32504
St Petersburg Jr. College	Clearwater Florida	33515
Abraham Baldwin Agric. College	Tifton, Georgia	31794
Black Hawk College	Moline, Illinois	61265
Chicago City College	Chicago, Illinois	60601
Danville Jr. College	Danville, Illinois	61832
Kaskaskia College	Centralia, Illinois	62801
Illinois Valley Cmty .College	La Salle, Illinois	61301
Morton Junior College	Cicero, Illinois	60650
Rock Valley College	Rockford, Illinois	61111
Southern Illinois Univ-VTI	Carbondale, Illinois	62901
Elgin Com. College	Elgin, Illinois	60120
Triton College	Northlake, Illinois	60164
Vincennes University	Vincennes Indiana	47591
Butler Country Cmty Jr College	El Dorado Kansas	67042
Eastern Kentucky University	Richmond, Kentucky	40475

Table 6 (cont)

Colleges Offering an Associate Degree
in
Data Processing

(Cont'd)

SCHOOL	LOCATION	ZIP CODE
Annevarundel Cmty. College	Arnold, Maryland	21012
Community College of Baltimore	Baltimore, Maryland	21299
Harford Jr College	Bel Air, Maryland	21014
Newton Jr. College	Newtonville, Massachusetts	02160
Delta College	Bay City, Michigan	48710
Ferris State College	Big Rapids, Michigan	49307
Flint Cmty Jr. College	Flint, Michigan	48503
Lansing Community College	Lansing, Michigan	48914
Macomb County Cmty College	Warren, Michigan	48093
Muskegon Co. Cmty College	Muskegon, Michigan	49442
Northwestern Michigan College	Traverse City, Michigan	49684
Schoolcraft College	Livonia, Michigan	48151
Washtenaw Cmty College	AnnArbor, Michigan	48107
Copiah Lincoln Jr. College	Wesson, Mississippi	39191
Jefferson Davis Jr. College	Gulfport, Mississippi	39501
Central Missouri St. College	Warrensburg, Missouri	64093
Florissant Valley Cmty. College	St. Louis, Missouri	63135
Meramec Cmty. College	St. Louis, Missouri	63122
Forest Park Cmty. College	St. Louis, Missouri	63110
Metro Jr. College K. C., Missouri	Kansas City, Missouri	64111
Missouri Southern College	Joplin, Missouri	64801
New Hampshire Technical Inst.	Concord, New Hampshire	03301
Ocean County College	Toms River, New Jersey	08753
Mercer County Community College	Trenton, New Jersey	08608
New Mexico Junior College	Hobbs, New Mexico	88240
New Mexico State Univ.	University Park, New Mexico	88001
Cuny Manhattan Cmty College	New York, New York	10020
Cuny Kingsboro Cmty College	Brooklyn, New York	11235
Cuny N. Y. City Cmty. College	Brooklyn, New York	11201
Genesee Cmty College	Batavia, New York	14020
Suny Ag. Tech. Alfred	Alfred, New York	14802
Suny Ag. Tech. Canton	Canton, New York	13617
Suny Ag. Tech. Cobleskill	Cobleskill, New York	12043
Suny Ag. Tech. Farmingdale	Farmingdale, New York	11735
Suny Ag. Tech. Morrisville	Morrisville, New York	13408
Auburn Comm. College	Auburn, New York	13021
Dutchess Cmty. College	Poughkeepsie, New York	12601
Erie Co. Tech. Institute	Buffalo, New York	14221
Hudson Valley Cmty. College	Troy, New York	12180
Monroe Community College	Rochester, New York	14607
Nassau Community College	Garden City, New York	11530
Orange County Cmty. College	Middletown, New York	10940
Suffolk Cmty College	Selden, New York	11784
Rockingham Cmty. College	Wentworth, North Carolina	27375
Western Carolina University	Cullowhee, North Carolina	28723

Table 6 (cont)

Colleges Offering an Associate Degree
In
Data Processing

(Cont'd)

SCHOOL	LOCATION	ZIP CODE
Bismarck Jr. College	Bismarck, North Dakota	58501
North Dakota State Sch. of Sci.	Wahpeton, North Dakota	58075
Cuyhog Cmty. College Metro Cam.	Cleveland, Ohio	44115
Sinclair Cmty. College	Dayton, Ohio	45402
Central Oregon Cmty. College	Bend, Oregon	97701
Harrisburg Area Cmty. College	Harrisburg, Pennsylvania	17199
Montgomery C. Cmty. College	Conshohocken, Pennsylvania	19248
Rhode Island Jr. College	Providence, Rhode Island	02908
Greenville Tech.	Greenville, South Carolina	29606
Richland Technical Ed. Cir.	Columbia, South Carolina	29205
Chattanooga State Tech. Inst.	Chattanooga, Tennessee	37406
Cisco Junior College	Cisco, Texas	76437
Cooke County Jr. College	Gainesville, Texas	76240
Dallas Cty. Jr. College Dist.	Dallas, Texas	75202
Del Mar College	Corpus Christi, Texas	78404
Grayson Co. Jr. College	Denison, Texas	75020
Navarro Jr. College	Corsicana, Texas	75110
Odessa College	Odessa, Texas	79760
San Antonio College	San Antonio, Texas	78212
Texarkana College	Texarkana, Texas	75501
Wharton County Jr. College	Wharton, Texas	77488
Weber State College	Ogden, Utah	84403
Northern Va. Cmty. College	Bailey Cross Road, Virginia	22041
Richmond Prof. Institute	Richmond, Virginia	23220
Everett College	Everett, Washington	98201
Grays Harbor College	Aberdeen, Washington	98520
Highline College	Midway, Washington	98031
Seattle Community College	Seattle, Washington	98122
Spokane Community College	Spokane, Washington	99202
U. Puerto Rico Rio Piedras	Rio Piedras, Puerto Rico	00931
U. Puerto Rico Mayaguez	Mayaguez, Puerto Rico	00708

Table 6 (cont)

Colleges Offering an Associate Degree
in
Computer Science

SCHOOL	LOCATION	ZIP CODE
Cadsden State Jr. College	E. Gadsden, Alabama	35903
Jefferson State Jr. College	Birmingham, Alabama	35215
Cornell University	Ithaca, New York	14850
Youngstown State University	Youngstown, Ohio	44503
Southwestern State College	Weatherford, Oklahoma	73096
Chattanooga State Tech. Institute	Chattanooga, Tennessee	37406
Columbia State Cmty. College	Columbia, Tennessee	38401

Colleges Offering an Associate Degree
in
Computer Programming

SCHOOL	LOCATION	ZIP CODE
Brevard Jr. College	Cocoa, Florida	32922
Manatee Jr. College	Bradenton, Florida	33505
Bristol Cmty. College	Fall River, Massachusetts	02720
Potomac St. College of W. Va. U.	Keyser, West Virginia	26726

Colleges Offering an Associate Degree
in
Computer Technology

SCHOOL	LOCATION	ZIP CODE
Prairie State College	Chicago Heights, Illinois	60411
University of Evansville	Evansville, Indiana	47704
Montgomery JC Takoma Park	Takoma Park, Maryland	20012
Montgomery JC Rockville	Rockville, Maryland	20850
New York Institute of Tech.	New York, New York	10023
Voorhees Technical Institute	New York, New York	10036
Ohio College of App. Science	Cincinnati, Ohio	45210
Allegheny Cmty College	Pittsburgh, Pennsylvania	15212

Table 6 (cont)

Colleges Offering an Associate Degree
in
Quantitative Methods

SCHOOL	LOCATION	ZIP CODE
College of St. Thomas	St. Paul, Minnesota	55101

Colleges Offering an Associate Degree
in
Opt. in Eng.

Oregon Technical Institute	Klamath Falls, Oregon	97601

6. LIST OF ATTENDEES

Prof. Richard Andree
Dept. of Mathematics
University of Oklahoma
Norman, Oklahoma 73069

Dr. Bruce W. Arden
Associate Director
Computing Center
University of Michigan
Ann Arbor, Michigan

Prof. Thomas H. Bredt
Dept. of Electrical Engineering
Stanford University
Stanford, California 94305

Dr. John Carr, III
Moore School of Engineering
Dept. of Computer Science
University of Pennsylvania
Philadelphia, Pennsylvania

Dr. C.L. Coates
Electronics Research Center
University of Texas at Austin
Austin, Texas 7871s

B.H. Colvin
Head, Mathematics Research Laboratory
Boeing Scientific Research Laboratories
P.O. Box 3981
Seattle, Washington 98124

Mr. Kent Curtis
Office of Computing Activities
National Science Foundation
Washington, D.C.

Dr. Ruth Davis
National Institutes for Health
National Library of Medicine
Bethesda, Md.

Prof. George E. Forsythe
Computer Science Department
Stanford University
Stanford, California 94305

Mrs. Alexandra Forsythe
Gunn High School
Palo Alto, California 94305

Dr. John Giese
Chief, Applied Mathematics Division
Department of the Army
U.S. Army Ballistic Research Laboratories
Aberdeen Proving Ground, Maryland 21005

Mr. Bruce Gilchrist
Executive Director
American Federation of Information Processing Societies
210 Summit Ave.
Montvale, N.J. 07645

Prof. J.W. Graham
Computing Centre Director
University of Waterloo
Waterloo, Ontario, Canada

Prof. Fred Gruenberger
Department of Accounting
San Fernando Valley State College
Northridge, California 91324

Dr. John W. Hamblen
Southern Regional Education Board
130 6th Street N.W.
Atlanta, Georgia 30313

Prof. Juris Hartmanis
Department of Computer Science
Cornell University
Ithaca, New York

Dr. Walter W. Jacobs
1812 Metzerott Road
Adelphi, Maryland 20783

Dr. T.L. Jordan
University of California
Los Alamos Scientific Laboratory
P.O. Box 1663
Los Alamos, New Mexico 87544

Prof. Edward J. McCluskey
SEL Digital Systems Laboratory
Stanford University
Stanford, California 94305

Prof. William F. Miller
Computer Science Department
Stanford University
Stanford, California 94305

Mr. Scott E. Moore
Manager of SDD Technical Education
IBM Systems Development Division
Department H77, Building 962
Box 390
Poughkeepsie, New York 12602

Mr. Robert Morris
Bell Telephone Laboratories, Inc.
Room 2C-524
Mountain Avenue
Murray Hill, New Jersey 07974

Prof. Alan J. Perlis
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Saul Rosen, Director
Computer Sciences Center
Mathematical Sciences Building
Purdue University
Lafayette, Indiana 47907

Mr. James Rowe
Union Carbide
270 Park Avenue (41st floor)
New York, New York

Dr. Samuel Seely
Associate Graduate Dean
University of Massachusetts
Amherst, Massachusetts 01002

Prof. J.N. Snyder
Associate Head of Computer Science
University of Illinois
Urbana, Illinois 61801

Dr. Robert Spinrad
Scientific Data Systems
701 South Aviation Boulevard
El Segundo, California 90245

Prof. John W. Tukey
Department of Statistics
Fine Hall, P.O. Box 708
Princeton, New Jersey 08540

SIC Report

The Data Base Panel
of the
Computer Science and Engineering Board
of the
National Academy of Sciences

PROPOSED CHANGES TO THE STANDARD INDUSTRIAL CLASSIFICATION

ABOUT THE SIC

The U. S. Standard Industrial Classification (SIC)* is used by all Government agencies (and by many private concerns, such as Dun and Bradstreet) as a common framework for collecting and reporting industrial, commercial, and economic statistics. The SIC provides a four-digit industry code for identifying the major economic activity of any "establishment," which, to simplify data reporting and collecting, is vaguely defined and need not necessarily correspond to any organizational or geographic entity.

Related industries are organized into groups, which are distinguished by the first three digits of their SIC code. Related groups are organized into major groups, distinguished by the first two digits of the code, and related major groups are organized into the following divisions: (a) agriculture, forestry, and fisheries; (b) mining; (c) contract construction; (d) manufacturing; (e) transportation, communication, and utilities; (f) wholesale and retail trade; (g) finance, insurance, and real estate; (h) services; (i) government; (j) nonclassifiable establishments.

Since the digit 0 is used to identify an unknown establishment (where not enough information is available to completely classify it), and since the digit 9 is used to identify miscellaneous categories, a group may contain no more than eight, non-miscellaneous industries, and a major group no more than eight non-miscellaneous groups.

An inter-agency Technical Committee on Industrial Classification, chaired by Milo Peterson of the Bureau of the Budget, is now engaged in revising the SIC. They will consider proposed revisions from any source, if accompanied by documented justification, until June 30, 1970. The revised edition of the classification will be effective January 1, 1972.

* Published in the "Standard Industrial Classification Manual, 1967," available from the Government Printing Office for $4.50 a copy.

According to the criteria established by this Committee, a proposed new industry should be significant, specialized, and inclusive. A new industry is considered to be significant if it has at least 20 percent of the number of establishments, employees, and dollar-value-added of the average industry in the same division. It is considered to be specialized if at least 80 percent of the products produced by establishments in the industry are those defining the industry. And it is considered to be inclusive if at least 70 percent of the products defining the industry (50 percent where these products are produced in significant amounts in other industries for internal use) are produced by establishments in the industry.

Still another criterion is comparability. A new classification that involves only the simple amalgamation or division of current industries facilitates comparisons with statistics kept under the old classification, and is thus preferred to a classification wherein new industries are made up of bits and pieces of several current industries. The fragmentation of miscellaneous industries and groups is presumably acceptable, however.

COMPUTING ITEMS IN CURRENT SIC MANUAL

Establishments in what is commonly called the computing industry are currently identified by the following codes:

2645	Die Cut Paper and Paperboard and cardboard (which includes tabulating card manufacturing)
3573	Electronic Computing Equipment
7392	Business, Management, Administrative and Consulting Services (which includes computer programming services)

7394	Equipment Rental and Leasing Services (which includes electronic equipment rental and leasing)
8242	Vocational Schools (which includes data processing schools)
8931	Accounting, Auditing, and Bookkeeping Services (which includes data processing services)

The Panel proposes that the Board recommend changes to the SIC that will better reflect both the structure and the importance of the computing industry as it now exists and as it is likely in the next few years to become.

The Panel's goals in this are two-fold: (1) that the revised SIC permit the separate identification of all the various sub-industries that make up the computing industry and (2) that it reflect the present and near future structure of the computing industry in as much detail as is consistent with the established criteria.

The Panel's proposed change to the SIC calls for the segments of the computing industry, with a few necessary exceptions, to be concentrated in two Major Groups -- one, Information Processing Equipment and Supplies, within the Manufacturing Division, (which would replace Group No. 357: Office, Computing, and Accounting Machines), the other, Information Processing Products and Services, a new Major Group within the Services Division.

The data in support of the proposal was prepared by the International Data Corporation at the request of the panel. The following comments by IDC indicate the significance of the data.

The estimates for 1969 are based on an extensive survey and analysis of the expenditures for computer-related products and services by users of computers and data processing equipment in the United States. We have estimated exports of computer-related equipment and services in order to provide an estimate of the total value of equipment and services produced by establishments within the United States. The preliminary estimates were checked against statistics on the computer installation census file maintained by IDC, from various trade reports, government statistics, and related information.

The counts for the number of establishments were estimated from an analysis of trade directories, association officers, annual reports and related references. There establishment was defined as a single physical location engaged in the identified activity. Generally it was a plant, office, or local service facility. Particularly in the case of the equipment manufacturing sectors, each of the principal firms engaged in the designated industry have several to over 50 establishments.

The "number of employees" estimate was prepared from an analysis of questionnaires received from firms designed in industry, trade directories, annual reports, government statistics and related information. An employee was counted as participating in an industry if he spent at least 25% of his time engaged in the activities of that industry. There is, therefore, considerable double-counting of employees in the computer systems/peripheral equipment sector where administrative and marketing personnel support the administration and sales of heterogeneous product lines.

Likewise, in the proprietary software/programming/systems analysis sector, many of the management, marketing, and technical personnel are involved simultaneously with two or three of these activities on a continuing basis.

The forecasts for 1974 are made by combining projections of customer demands established by customer requirement studies, investigation of the untapped potential of the user computer systems, analysis of the contributions of new information technology to the growth of computer applications, a projection of the economic environment under which the sales of computer related products and services will take place. Because of the high levels of uncertainty involved in each of the elements of this projection, the figures must be considered highly speculative. This is particularly true in the measurement of the number of establishments.

Proposed SIC No.	Proposed Industry Description	Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports)					
		Size Measures					
		1969			1974		
		No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)	No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)
X	INFORMATION PROCESSING EQUIPMENT AND SUPPLIES						
3X1	COMPUTERS						
3X11	Digital Computers (Value relates to Central Processor and internal core storage only)	375	148,000	2260	350-420	160,000	3300
3X19	Miscellaneous Computers, including Analog and Hybrid Computers (not elsewhere classified)						
	Estimates for Analog computers	40	2,800	52	35-45	1,350	30
3X2	COMPUTER PERIPHERAL EQUIPMENT						
3X21	Punched Card Handling Equipment (on and off-line)	30	100,000	700	40-50	150,000	920
3X22	Printers	40	95,000	650	50-60	105,000	885
3X23	Optical and Magnetic Character Readers and Writers	35	20,000	110	55-65	50,000	260
3X24	Display Equipment (including graphic display, plotters, and interactive line and character displays)	25	18,000	40	70-85	50,000	165

roposed IC o.	Proposed Industry Description	1969 No. of Estab.	1969 Total Employ.	1969 Value of Shipments or Services ($ Mill.)	1974 No. of Estab.	1974 Total Employ.	1974 Value of Shipments or Services ($ Mill.)
3X25	Auxiliary Storage Equipment (magnetic tape drives, disk files, disk pack drives, magnetic card and strip readers, magnetic drums, and related equipment.)	45	130,000	1650	68-85	155,000	2600
3X26	Computer Terminals (Including Conversational and Remote Batch, except purely communication terminals)	140	96,000	220	250-300	120,000	960
3X29	Miscellaneous						
3X3	ADDING MACHINES, DESK CALCULATORS, ACCOUNTING MACHINES AND TABULATING MACHINES (Now listed as 3574)						
3X4	TYPEWRITERS (Now listed as 3572)						
3X5	COPYING AND DUPLICATING MACHINES (Now listed under 3579)						
3X6	MICROFORM EQUIPMENT						
3X8	INFORMATION PROCESSING EQUIPMENT SUPPLIES AND ACCESSORIES	450	48,000	880	500-600	70,000	1410

Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports) — Size Measures

Proposed SIC No.	Proposed Industry Description	1969 No. of Estab.	1969 Total Employ.	1969 Value of Shipments or Services ($ Mill.)	1974 No. of Estab.	1974 Total Employ.	1974 Value of Shipments or Services ($ Mill.)
3X9	MISCELLANEOUS INFORMATION PROCESSING EQUIPMENT						
3X91	Mailing and Addressing Equipment (Now listed as 3579)						
3X92	Scales and Balances, Except Laboratory (Now listed as 3574)						
3X99	Information Processing and Office Equipment, Not Elsewhere Classified.						

Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports) — Size Measures

Proposed SIC No.	Proposed Industry Description	1969 No. of Estab.	1969 Total Employ.	1969 Value of Shipments or Services ($ Mill.)	1974 No. of Estab.	1974 Total Employ.	1974 Value of Shipments or Services ($ Mill.)
	Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports) — Size Measures						
74	INFORMATION PROCESSING PRODUCTS AND SERVICES						
741	COMPUTER PROGRAMS AND PROGRAMMING SERVICES						
7411	Proprietary Computer Programs (Computer programs designed to be sold with no or a minimum amount of customization to the individual customer.)	200	2,500	20	700-900	100,000	1550
7412	Computer Programming Services (custom contract services in programming and coding computer instructions, including work done on an hourly or daily rate basis.)	1000	21,000	360	2000-3000	110,000	1335
7413	Systems Analysis, Design, Evaluation, Selection and Consulting Services (Limited to those directly involved with the application of computer systems.)	300	3,800	65	1500-2000	110,000	625
7419	Computer Programs and Programming Services (Not Elsewhere Classified)						

oposed C	Proposed Industry Description	Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports)					
		Size Measures					
		1969			1974		
		No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)	No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)
742	COMPUTATIONAL AND ALLIED SERVICES						
7421	Computing Services	2500	28,000	860	3800-5000	130,000	1800
7422	Data Preparation and Conversion Services	2800	12,000	80	3000-4000	28,000	175
7423	Computer Facility Management and Turnkey System Development	30	1,200	20	200-300	8,000	100
7429	Miscellaneous Computational Services (Not Elsewhere Classified)						
743	INFORMATION PROCESSING EQUIPMENT REPAIR AND MAINTENANCE	400	40,000	500	700-800	75,000	1100
744	INFORMATION PROCESSING EQUIPMENT RENTAL AND LEASING AND USED EQUIPMENT SALES	250	5,000	330	350-400	10,000	700
745	ACCOUNTING, AUDITING, AND BOOKKEEPING SERVICES (replaces current Group No. 893)						
749	MISCELLANEOUS INFORMATION PROCESSING PRODUCTS AND SERVICES (NOT ELSEWHERE CLASSIFIED)						

In addition to these two major groups, the Panel recommends one new category; namely,

7362 Information Processing Employment Agencies

and the addition of,

Data Processing Vocational Schools
(under Group No. 8242)

within Group No. 736 (Private Employment Agencies) and Group No. 824 (Vocational Schools, Except Vocational High Schools) respectively.

Day Agenda

COMPUTER SCIENCE AND ENGINEERING BOARD

AGENDA

Day Session 5 May 1970

Philadelphia International Airport Motel

0900 - 1000	Roger Levien The Rand Corporation "Instructional Uses of the Computer in Higher Education"
1000 - 1200	Dr. Alan J. Perlis Draft IV of Summer Conference Report (Three Brazilian visitors will be attending. They are guests of the Office of Foreign Secretary, NAS).
1200 - 1300	Lunch, Brass Rail, Philadelphia International Airport
1300 - 1400	Dr. Alan J. Perlis (Continued from the morning)
1400 - 1600	FCC Report
1600 - 1630	Auerbach
1630 - 1700	Six Month Meeting/Activity Projection
1700 - 1800	Computer Priorities Summer Conference
1800 - 1900	Dr. Alan Westin and Prof. E. Dial "Questionnaire"
1900 - 1920	Terms of Board Membership

Levien - Outline

A-60

INSTRUCTIONAL USES OF THE COMPUTER IN HIGHER EDUCATION

- Revised Outline -

R. E. Levien

S. Barro
F. Blackwell
G. Comstock
K. Hoffmayer
W. Holland
C. Mosmann

The Rand Corporation
November 1969

Section I

INTRODUCTION TO THE COMPUTER IN HIGHER EDUCATION

Chap. I. Introduction

A description of major trends in computer technology and applications in general and their relevance to higher education.

Chap. II. Research Uses of Computers

A. Research on Computers

The character of hardware, software, and computer theory research on campus and in industry.

B. Research with Computers

The numeric, symbolic, data, sign, and event processing capabilities of computers applied in physical, social, biological sciences, humanities, and arts.

Chap. III. Administrative Uses of Computers

The computer's uses in college and university bookkeeping, record-keeping, and planning.

Chap. IV. Instructional Uses of Computers

A. Instruction About Computers

The need to teach computer specialists, computer users, and the public about computers.

B. Instruction with Computers

A description of the many ways the computer can assist the performance of instruction, the management of instruction, and their combination into fully computer-based instruction.

Chap. V. Computer Services on Campus

A. Provision of Computer Services

Problems of bringing computer power onto campus and descriptions of alternative methods of doing so.

B. Use of Computer Services

Problems of allocating and applying computer power on campus and descriptions of alternative solutions.

Appendix A. An Introduction to Computers

A concise summary of the basic concepts and vocabulary of computing.

Appendix B. Scenarios of Instructional Uses

Several descriptions of effective uses of the computer in instruction in different subjects: music, physics, literature, medicine, languages, chemistry.

Section II

THE CURRENT STATE OF INSTRUCTIONAL USES OF THE COMPUTER

Chap. I. Introduction

A summary of general trends in the computer's instructional use in higher education today.

Chap. II. Utilization of Computers---National

A characterization, based on several recent national surveys, of the state of computer use for instruction in the almost 2500 institutions of higher education in the United States. The description is in terms of major categories: public and private universities, four-year colleges, and two-year colleges.

Chap. III. Utilization of Computers--California

A more detailed characterization, based on a special Rand survey of the 200 or so California higher educational institutions, of the nature of instructional uses in the several categories of institution. Special attention is paid to perceived impediments and attitudes.

Chap. IV. Technology of Computers

A. Hardware

Current state of processor, storage, peripheral, and communication devices.

B. Software

Current state of system and application programs and languages.

C. Systems

Current state of hardware-software combinations for time-sharing, batch processing, and control.

D. Costs

Current costs of hardware, software, systems.

B. Hardware

Forecasts of performance and costs and their dependence on support for R and D.

C. Software

Forecasts of performance and production costs and their dependence on marketing and distribution arrangements.

D. Systems

Forecasts of performance and costs and their dependence on institutional arrangements.

E. Alternative Modes of Instruction

Forecasts of performance and costs for alternative modes of instruction, subjects, and levels of instruction.

F. Financing Instruction

Impact of alternative financing arrangements on the adoption of computer uses in instruction.

Chap. III. Organizational and Institutional Factors

Impact of alternative institutions for providing computer service and producing and distributing materials on the future of instructional uses. Impact of computer uses on the organization of colleges and universities.

Chap. IV. Acceptability Factors

Problems in gaining acceptance by students, faculty, and administrators. Programs for improving acceptability.

Chap. V. Alternative Futures

A description of several possible 10- to 20-year development possibilities for instructional uses of the computer. Discussion of alternative technological developments, institutional arrangements, development strategies, and financial arrangements.

Chap. VI. Recommendations for Action

Suggestions for activities by colleges and universities, government, industry, foundations, and other interested parties.

9:00 am
MAY 4 REC'D
Olsen

Battelle Memorial Institute · WASHINGTON, D.C. OFFICES

1755 MASSACHUSETTS AVENUE, N.W., WASHINGTON, D.C. 20036 · AREA CODE 202, TELEPHONE 232-8553

April 30, 1970

Mr. Warren House
National Academy of Science
21st & Pennsylvania Ave., N.W.
Washington, D. C.

Dear Warren:

Attached is a very brief and informal presentation paper on CAD/CAM.

I would appreciate a chance to call you at some convenient time during your Monday session. Panel 14 will have met by then and I may have some additional information on the objectives of this group by Monday noon, May 4th.

Thank you for your interest in this matter.

Shifted to Tuesday.

Very truly yours,

Geo Beiser

GEORGE BEISER
Senior Technical Advisor

GB:cap

Attachments

PRESENTATION PAPER ON CAD/CAM

The increasing interest of industry and government agencies -- particularly the Department of Defense -- in Computer Aided Design and Computer Aided Manufacturing (CAD/CAM) has reached a point where a central authoritative office is needed to serve as a national coordinating center. Specifically, a group of individuals from industry that has been working on a wide variety of CAD/CAM problems would like to explore possible affiliation with the Computer Research & Engineering Board to continue their work on such topics as new technology, education and management in this subject area.

The background on the above development involves the gathering of a group of specialists to assist the Department of Defense in putting on the national CAD/CAM conference that was held at Davenport, Iowa, during October of 1969. This group was organized into panels, 12 of which covered different aspects of CAD/CAM operations. Panel 13 summarized the findings of the twelve panels and Panel 14 assumed the responsibility for preparing recommendations that would benefit the national CAD/CAM program. One of these recommendations expresses the need for a national coordinating center.

Panel 14 is the only panel that is still active. It is made up of two representatives from each of the following engineering societies and associations: National Security Industrial Association, American Ordnance Association, Electronic Industries Association, Aerospace Industries Association, the Society of Manufacturing Engineers, and the Numerical Control Society. This group has worked well together for many months and has been quite effective in achieving its immediate objectives. It now feels the need for communicating and planning with a knowledgeable permanent group on a series of problems which require further attention. These problems exist in areas such as training, languages, standards, contracting, etc.

Panel 14 will meet on 1 & 2 of May at Cocoa Beach, Florida, to further consider selected problems that it believes warrant immediate attention. If your reaction to this inquiry is favorable, plans could be made for more formal discussions and presentations at a future meeting of the Board.

[illegible]

Olsen
9:00 2v
MAY 4 Rec'd

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D.C., 20418

27 April 1970

ANTHONY G. OETTINGER, CHAIRMAN
COMPUTER SCIENCE & ENGINEERING BOARD
AIKEN COMPUTATION LABORATORY
HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS 02138

TO: Warren C. House

Dear Warren,

This follows up on your memo dated April 20 to Sid Fernbach regarding relations with NSF.

You did mention Chris Schubert's interest in the Board's looking over the proposed AFIPS study but you asked me in front of Bruce Gilchrist and I felt constrained!

Especially now after the episode concerning international affairs, I am most reluctant for us to give the appearance of endorsing an AFIPS study from the AFIPS side. Doing it as participants in an interagency group of the type Chris Schubert proposes is a horse of an entirely different color and I should think that the Executive Secretary of the Board might be quite appropriate for the observer, especially if he stays in very close touch with the Chairman of the Data Base Panel!

Furthermore, the idea of a general relationship with the NSF in the Computer field is attractive, the moreso following upon the interview Sid and I had with John Pasta, which left both of us worried about the future of the Office of Computing Activities.

You will note that our contracts to date with NSF have come through OAC and perforce we must continue to work with them to some degree. I think, however, that our mandate is broader than what is covered by the OAC backwater and the general relationship might better be to the National Science Board, the Director of NSF, or one of the new assistant Directors.

Let us put this matter up for brief discussion at the May meeting. Depending on the Board's views, the next step might well be a chat with the Chairman of COSPUP and the President of the Academy, who, as you all know, also wear National Science Board hats.

Sincerely yours,

[illegible]

Anthony G. Oettinger

AGO:chm

cc: S. Fernbach, J. Pierce, J. Griffith

Olsen

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D.C., 20418

9:00 Jo
MAY 4 REC'D

28 April 1970

ANTHONY G. OETTINGER, CHAIRMAN
COMPUTER SCIENCE & ENGINEERING BOARD
AIKEN COMPUTATION LABORATORY
HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS 02138

TO: Dr. Walter S. Baer

Dear Wally,

Your letter of April 20 makes much sense to me. I am therefore taking the liberty to pass it on to both Alan Perlis and Warren House along with a copy of this reply.

I'd like to take this up at our May meeting. Perhaps Alan Perlis could pull out of his report a one or two page statement of this particular matter for review by the Board. If the Board agrees with the proposition, we might then follow through in the manner that you suggest.

Our proceeding in this way would have the merit of not tying this particular matter to the process of passing the new draft of the report itself through the hands of the participants at the Annapolis meeting.

Sincerely yours,

Tony

Anthony G. Oettinger

AGO:chm

cc: ~~W.~~ House
A. Perlis
J. Pierce

LAIRD
SYSTEMS
INCORPORATED

April 20, 1970

Professor Anthony Oettinger
Aiken Computation Laboratory
Harvard University
Cambridge, Massachusetts

Dear Tony:

I think one comment on the Perlis Report deserves restating. The report mentions the possibility of retraining physicists in computer science, both to satisfy the demand for well-trained computer scientists and to reduce the current over-supply of PhD physicists. While this is made a minor recommendation in the Panel report, I believe that it should be emphasized by the CS&EB if it has merit.

Short-term changes in supply and demand among highly trained scientists and engineers is a serious national problem which the Federal government has not adequately addressed. While there have been attempts by people within BOB and OST (including myself) to encourage retreading, there is today practically no support that I know of for retraining scientists who wish to change fields after receiving their doctorates or in mid-career. The physics-to-computer science transfer might provide a good model for a small scale, experimental federal support program. The problem is most relevant, and a program probably could be started relatively quickly with relatively little money. NSF would seem a likely source of funds.

If after further examination this appears to be a feasible recommendation, I believe the Board should emphasize it strongly and push for its implementation.. If presented this Spring, the idea easily could be put into the FY'72 budget next Fall. A presentation to BOB, OST, and NSF would seem a logical next step.

Sincerely,

[illegible]

Walter S. Baer

NATIONAL ACADEMY OF SCIENCES

COMPUTER SCIENCE & ENGINEERING BOARD
2101 CONSTITUTION AVENUE
WASHINGTON, D. C. 20418

COMPUTER SCIENCE AND ENGINEERING BOARD

JUNE 17, 18

Burchard
Jack Kettler

ATTENDANCE LIST

ATTENDEES

Professor Anthony G. Oettinger

Dr. Walter S. Baer
Dr. Launor F. Carter
Prof. Wesley A. Clark
Dr. Sidney Fernbach
Dr. Martin Greenberger (17th only)
Mr. Jerrier A. Haddad
Mr. William T. Knox
Dr. J. C. R. Licklider
Dr. John R. Meyer
Prof. William F. Miller
Mr. Roy Nutt
Mr. Kenneth Olsen
Dr. Alan J. Perlis
Prof. Barkley Rosser
Dr. Ronald L. Wigington

Consultants

Mr. John Griffith
Dr. Bernhard Romberg

Guests for Morning of 18th

Mr. Robert Courtney - IBM
Mr. Edward Glaser - Case Western Reserve University
Mr. George Hicken - National Security Agency
Mr. James Anderson - Consultant to National Security Agency
Mr. Ronald Kaufman - Security Officer of National Security Agency
Mr. Clark Weissman - SDC

Guests for Afternoon of 18th

Mr. Isaac Auerbach - Member, IFIPS
Mr. Fred Hayes - Bureau of the Budget, NYC (tentative)

Page 2
Attendance List
Computer Science & Engineering Board
June 17, 18

<u>Guests for Afternoon of 18th</u>

Mr. Robert Bruce, Bureau of the Budget, NYC
Mr. James Masters, Bureau of the Budget, NYC
Mr. George Beiser, Batelle Memorial Institute

ABSENTEES

Board Members

Mr. William L. Lurie
Dr. John R. Pierce
Dr. Alan F. Westin

Consultant

Mr. Joel Coehn

Executive Session

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D.C. 20418

COMPUTER SCIENCE AND ENGINEERING BOARD

Executive Session June 17, 1970

Joseph Henry Building

7:30 P.M. Room 500A

Panel Status Review and General Discussion

FCC	Dr. A. G. Oettinger
Privacy	Dr. A. G. Oettinger
Reorientation of Computer Export Panel	Dr. S. Fernbach/Dr. Oettinger
Information Systems Panel	Dr. Ronald L. Wigington
Summer Conference on Use of Computers in Colleges & Universities in the Coming Decade	Dr. A. G. Oettinger
Science and Public Policy	Mr. Kurt Borchardt

Review of Six-month Schedule	Dr. A. G. Oettinger

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D. C. 20418

COMPUTER SCIENCE & ENGINEERING BOARD

Executive Session June 17, 1970

Joseph Henry Building

7:30 P. M.

500 A

Panel Status Review and General Discussion

FCC	Mr. Lewis S. Billig
Privacy	Dr. Alan F. Westin
Reorientation of Computer Export Panel	Dr. Fernbach/Dr. Oettinger
Information Systems Panel	Dr. Ronald L. Wigington
Summer Conference on Use of Computers in Colleges & Universities in Coming Decade	Dr. A. G. Oettinger
Science and Public Policy	Mr. Kurt Borchardt

Review of Six-month Schedule

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D. C. 20418

COMPUTER SCIENCE & ENGINEERING BOARD

Day Session June 18, 1970

NATIONAL ACADEMY OF SCIENCES
21ST & Pennsylvania Avenue, N. W.
Joseph Henry Building
Room 500A

Warren C. House, Executive Secretary
Tel: 202-961-1386

MORNING SESSION - CLASSIFIED

0900 - 0930	Robert Courtney IBM Corporation Dept. D88, Building 931 P. O. Box 390 Poughkeepsie, N. Y. 12602 Tel: 914-485-8887 The state of the art in computer security in general, the emerging requirement for secure computer operations, and probable developments to meet such requirements. TOP SECRET
0930 - 1000	Edward Glaser Director of Computing Center Case-Western Reserve University Cleveland, Ohio 44100 Tel: 216-368-2808 The recent RAND study of the general problem of computer security. TOP SECRET
1000 - 1030	Willis Ware The Rand Corporation 1700 Main Street Santa Monica, California 90406 Tel: 213-393-0411 The recent RAND study of the general problem of computer security (tentative due to possible conflict with meeting with Air Force in Dayton, Ohio) TOP SECRET

Page 2, continued
June 18, 1970
Morning Session - CLASSIFIED

1030 - 1100	Clark Weissman System Development Corporation 2500 Colorado Avenue Santa Monica, California 90406 Tel: 213-393-9411, x533/534 Summary of two recent papers on computer security and general comments. TOP SECRET
1100 - 1130	George Hicken National Security Agency Room 1M070 Ft. George E. Meade, Maryland 20755 Tel: 301-688-7757 Computer network security and related developments. TOP SECRET
1130 - 1200	LUNCH (U)

Notes to Briefers: (1) The above are merely guidelines. Please modify the presentation order as you wish with others concerned. The order can be altered up to the last minute as necessary. The tight timing is subject to modest changes at this time.

(2) A topic outline of the general computer security problem will be distributed at the meeting.

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D.C. 20418

COMPUTER SCIENCE & ENGINEERING BOARD

Day Session June 18, 1970

Joseph Henry Building

Room 500A

<u>AFTERNOON SESSION - UNCLASSIFIED</u>

1200 - 1230	Dr. Sidney Fernbach Standard Industrial Classification
1230 - 1300	Mr. Issac Auerbach Planning Group Report on International Computer Consideration
1300 - 1330	Dr. Alan J. Perlis Report Review----Computer Science Education
1330 - 1500	Mr. Fred Hayes, Head, Bureau of the Budget, NYC and Mr. Robert Bruce, Bureau of the Budget, NYC Project Consultation
1500 - 1530	Mr. William Knox and Dr. Walter Baer Graduate Level Retraining into Computer Science & Engineering - a Proposal
1530 - 1545	Status of FCC Interconnections Report
1545 - 1630	Plans for Summer Conference Role of the Computer in Colleges and Universities in the Coming Decade

JUN 3 1970

NATIONAL ACADEMY OF SCIENCES

COMPUTER SCIENCE & ENGINEERING BOARD
2101 CONSTITUTION AVENUE
WASHINGTON, D. C. 20418

COMPUTER SCIENCE & ENGINEERING BOARD

June 17, 18

ATTENDANCE LIST

ATTENDEES

Professor Anthony G. Oettinger, Chairman

Dr. Walter S. Baer
Dr. Launor F. Carter
Professor Wesley A. Clark
Dr. Sidney Fernbach
Dr. Martin Greenberger (17th only)
Mr. Jerrier A. Haddad
Mr. William T. Knox
Dr. J. C. R. Licklider
Mr. William L. Lurie
Dr. John R. Meyer
Professor William F. Miller
Mr. Roy Nutt
Mr. Kenneth Olsen
Dr. Alan J. Perlis
Dr. John R. Pierce
Professor J. Barkley Rosser
Dr. Ronald L. Wigington

Consultants

Mr. John Griffith
Dr. Bernhard Romberg

Guests for Morning of 18th

Mr. Robert Courtney - IBM
Mr. Edward Glaser - Case Western Reserve University
Mr. George Hicken - National Security Agency
Dr. Willis Ware - Rand (tentative)
Mr. Clark Weissman - SDC

Guests for Afternoon of 18th

Mr. Issac Auerbach - Member, IFIPS
Mr. Fred Hayes - Bureau of the Budget, NYC
Mr. Robert Bruce - Bureau of the Budget, NYC

ABSENTEES

Dr. Alan F. Westin

Consultant

Mr. Joel Cohen

Day Session - AM

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D. C. 20418

COMPUTER SCIENCE AND ENGINEERING BOARD

Day Session June 18, 1970

NATIONAL ACADEMY OF SCIENCES
21st & Pennsylvania Avenue, N. W.
Joseph Henry Building
Room 500A

Warren C. House, Executive Secretary, CS&EB
Tel: 202-961-1386

MORNING SESSION - CLASSIFIED

0900 - 0935	Robert Courtney IBM Corporation Dept. D88, Building 931 P. O. Box 390 Poughkeepsie, New York 12602 Tel: 914-485-8887 The state of the art in computer security in general, the emerging requirement for secure computer operations, and probable developments to meet such requirements. TOP SECRET
0935 - 1010	Edward Glaser Director of Computing Center Case Western Reserve University Cleveland, Ohio 44100 Tel: 216-368-2808 The recent RAND study of the general problem of computer security. TOP SECRET

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D.C. 20418

June 18, 1970
Morning Session - Classified

1010 - 1045	Clark Weissman System Development Corporation 2500 Colorado Avenue Santa Monica, California 90406 Tel: 2.3-393-9411, x533/534 Summary of two recent papers on computer security and general comments. TOP SECRET
1045 - 1120	George Hicken National Security Agency Room 1M070 Ft. George E. Meade, Maryland 20755 Tel: 301-688-7757 Computer network security and related developments. TOP SECRET
1120 - 1200	LUNCH (U)

Notes to Briefers:

(1) Above order and times are arbitrary. Change at will.

(2) A topic outline of the general security problem will be distributed at the meeting.

Day Session - PM

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D.C. 20418

COMPUTER SCIENCE AND ENGINEERING BOARD

Day Session June 18, 1970

Joseph Henry Building
Room 500A

AFTERNOON SESSION - UNCLASSIFIED

1200 - 1230	Dr. Sidney Fernbach Standard Industrial Classification
1230 - 1300	Mr. Isaac Auerbach Planning Group Report on International Computer Consideration
1300 - 1400	Dr. Alan J. Perlis Report Review ----Computer Science Education
1400 - 1430	Mr. George Beiser Computer Aided Design/Computer Aided Manufacturing (CAD/CAM)
1430 - 1530	Mr. William Knox and Dr. Walter Baer Graduate Level Retraining into Computer Science & Engineering - a Proposal
1530 - 1600	Foreign Secretary's Office, NAS Brazilian Computer Scientists
1600 - 1630	Mr. Robert Bruce, Bureau of the Budget, NYC Mr. James Masters, Bureau of the Budget, NYC Mr. Fred Hayes, Bureau of the Budget, NYC (tentative) Project Consultation

Computer Security

Hardware aspects of secure computing

by LEE M. MOLHO

System Development Corporation
Santa Monica, California

INTRODUCTION

It makes no sense to discuss software for privacy-preserving or secure time-shared computing without considering the hardware on which it is to run. Software access controls rely upon certain pieces of hardware. If these can go dead or be deliberately disabled without warning, then all that remains is false security.

This paper is about hardware aspects of controlled-access time-shared computing.* A detailed study was recently made of two pieces of hardware that are required for secure time-sharing on an IBM System 360 Model 50 computer: the storage protection system and the Problem/Supervisor state control system.[1] It uncovered over a hundred cases where a single hardware failure will compromise security without giving an alarm. Hazards of this kind, which are present in any computer hardware which supports software access controls, have been essentially eliminated in the SDC ADEPT-50 Time-Sharing System through techniques described herein.[2]

Analysis based on that work has clarified what avenues are available for subversion via hardware; they are outlined in this paper. A number of ways to fill these security gaps are then developed, including methods applicable to a variety of computers. Administrative policy considerations, problems in security certification of hardware, and hardware design considerations for secure time-shared computing also receive comment.

FAILURE, SUBVERSION, AND SECURITY

Two types of security problem can be found in computer hardware. One is the problem of hardware failure. This includes not only computer logic that fails by itself, but also miswiring and faulty hardware caused by improper maintenance ("Customer Engineer") activity, including CE errors in making field-installable engineering changes.

The other security problem is the cloak-and-dagger question of the susceptibility of hardware to subversion by unauthorized persons. Can trivial hardware changes jeopardize a secure computing facility even if the software remains completely pure? This problem and the hardware failure problem, which will be considered in depth, are related.

Weak points for logic failure

Previous work involved an investigation of portions of the 360/50 hardware.[1] Its primary objective was to pinpoint single-failure problem locations. The question was asked, "If this element fails, will hardware required for secure computing go dead without giving an alarm?" A total of 99 single-failure hazards were found in the 360/50 storage protection hardware; they produce a variety of system effects. Three such logic elements were found in the simpler Problem/Supervisor state (PSW bit 15) logic. A failure in this logic would cause the 360/50 to always operate in the Supervisor state.

An assumption was made in finding single-failure logic problems which at first may seem more restrictive than it really is: A failure is defined as having occurred if the output of a logic element remains in an invalid state based on the states of its inputs. Other failure modes certainly exist for logic elements, but they reduce to this case as follows: (1) an intermittent logic element meets this criterion, but only part of the time; (2) a shorted or open input will cause an invalid output state at least part of the time; (3) a logic element which exhibits excessive signal delay will appear to have an invalid output state for some time after any input transition; (4) an output wire which has been con-

*The relationship between "security" and "privacy" has been discussed elsewhere.[3,4] In this paper "security" is used to cover controlled-access computing in general.

nected to an improper location will have an invalid output state based on its inputs at least part of the time; such a connection may also have permanently damaged the element, making its output independent of its input. It should be noted that failure possibilities were counted; for those relatively few cases where a security problem is caused whether the element gets stuck in "high" or in "low" state, two possibilities were counted.

A situation was frequently encountered which is considered in a general way in the following section, but which is touched upon here. Many more logic elements besides those tallied would cause the storage protection hardware to go dead if they failed, but fortunately (from a security viewpoint) their failure would cause some other essential part of the 360/50 to fail, leading to an overall system crash. "Failure detection by faulty system operation" keeps many logic elements from becoming security problems.

Circumventing logic failure

Providing redundant logic is a reasonable first suggestion as a means of eliminating single failures as security problems. However, redundancy has some limits which are not apparent until a close look is taken at the areas of security concern within the Central Processing Unit (CPU). Security problems are really in control logic, such as the logic activated by a storage protect violation signal, rather than in multi-bit data paths, where redundancy in the form of error-detecting and error-correcting codes is often useful. Indeed, the 360/50 CPU already uses an error-detecting code extensively, since parity checks are made on many multi-bit paths within it.

Effective use of redundant logic presents another problem. One must fully understand the system as it stands to know what needs to be added. Putting it another way, full hardware certification must take place before redundancy can be added (or appreciated, if the manufacturer claims it is there to begin with).

Lastly, some areas of hardware do not lend themselves too easily to redundancy: There can be only one address at a time to the Read-Only-Storage (ROS) unit whose microprograms control the 360/50 CPU.[5,6] One could, of course, use such a scheme as triple-modular redundancy on all control paths, providing three copies of ROS in the bargain. The result of such an approach would not be much like a 360/50.

Redundancy has a specialized, supplementary application in conjunction with hardware certification. After the process of certification reveals which logic elements can be checked by software at low overhead, redundant logic may be added to take care of the remainder. A good example is found in the storage protection logic. Eleven failure possibilities exist where protection interrupts would cause an incorrect microprogram branch upon failure. These failure possibilities arise in part from the logic elements driven by one control signal line. This signal could be provided redundantly to make the hardware secure.

Software tests provide another way to eliminate hardware failure as a security problem. Code can be written which should cause a protection or privileged-operation interrupt; to pass the test the interrupt must react appropriately. Such software must interface the operating system software for scheduling and storage-protect lock alteration, but must execute in Problem state to perform its tests. There is clearly a tradeoff between system overhead and rate of testing. As previously mentioned, hardware certification must be performed to ascertain what hardware can be checked by software tests, and how to check it.

Software testing of critical hardware is a simple and reasonable approach, given hardware certification; it is closely related to a larger problem, that of testing for software holes with software. Software testing of hardware, added to the SDC ADEPT-50 Time-Sharing System, has eliminated over 85 percent of present single-failure hazards in the 360/50 CPU.

Microprogramming could also be put to work to combat failure problems. A microprogrammed routine could be included in ROS which would automatically test critical hardware, taking immediate action if the test were not passed. Such a microprogram could either be in the form of an executable instruction (e.g., TEST PROTECTION), or could be automatic, as part of the timer-update sequence, for example.

A microprogrammed test would have much lower overhead than an equivalent software test performed at the same rate; if automatic, it would test even in the middle of user-program execution. A preliminary design of a storage-protection test that would be exercised every timer update time (60 times per second) indicated an overhead of only 0.015 percent (150 test cycles for every million ROS cycles). Of even greater significance is that microprogrammed testing is specifiable. A hardware vendor can be given the burden of proof of showing that the tests are complete; the vendor would have to take the testing requirement into account in design. The process of hardware certification could be reduced to a design review of vendor tests if this approach were taken.

Retrofitting microprogrammed testing in a 360/50 would not involve extensive hardware changes, but some changes would have to be made. Testing microprograms would have to be written by the manu facturer; new ROS storage elements would have to be fabricated. A small amount of logic and a large amount of documentation would also have to be changed.

Logic failure can be totally eliminated as a security problem in computer hardware by these methods. A finite effort and minor overhead are required; what logic is secured depends upon the approach taken. If microprogram or software functional testing is used, miswiring and dead hardware caused by CE errors will also be discovered.

Subversion techniques

It is worthwhile to take the position of a would-be system subverter, and proceed to look at the easiest and best ways of using the 360/50 to steal files from unsuspecting users. What hardware changes would have to be made to gain access to protected core memory or to enter the Supervisor state?

Fixed changes to eliminate hardware features are obvious enough; just remove the wire that carries the signal to set PSW bit 15, for example. But such changes are physically identical to hardware failures, since something is permanently wrong. As any functional testing for dead hardware will discover a fixed change, a potential subverter must be more clever.

In ADEPT–50, a user is swapped in periodically for a brief length of time (a "quantum"). During his quantum, a user can have access to the 360/50 at the machine-language level; no interpretive program comes between the user and his program unless, of course, he requests it. Thus, a clever subverter might seek to add some hardware logic to the CPU which would look for, say, a particular rather unusual sequence of two instructions in a program. Should that sequence appear, the added logic might disable storage protection for just a few dozen microseconds. Such a small "hole" in the hardware would be quite sufficient for the user to (1) access anyone's file; (2) cause a system crash; (3) modify anyone's file.

User-controllable changes could be implemented in many ways, with many modes of control and action besides this example (which was, however, one of the more effective schemes contemplated). Countermeasures to such controllable changes will be considered below, along with ways in which a subverter might try to anticipate countermeasures.

Countermeasures to subversion

As implied earlier, anyone who has sufficient access to the CPU to install his own "design changes" in the

a fixed change would be discovered by even a simple software test infrequently performed. A user-controllable change, on the other hand would not be discovered by tests outside the user's quantum, and would be hard to discover even within it, as will become obvious.

The automatic microprogrammed test previously discussed would have a low probability of discovering a user-controllable hardware change. Consider an attempt by a user to replace his log-in number with the log-in number of the person whose file he wants to steal. He must execute a MOVE CHARACTERS instruction of length 12 to do this, requiring only about 31 microseconds for the 360/50 CPU to perform. A microprogrammed test occurring at timer interrupts—once each 16 milliseconds—would have a low probability of discovering such a brief security breach. Increasing the test rate, though it raises the probability, raises the overhead correspondingly. A test occurring at 16 *micro*second intervals, for example, represents a 15 percent overhead.

A reasonable question is whether a software test might do a better job of spotting user-controllable hardware changes. One would approach this task by attempting to discover changes with tests inserted in user programs in an undetectable fashion. One typical method would do this by inserting invisible breakpoints into the user's instruction stream; when they were encountered during the user's quantum, a software test of storage protection and PSW bit 15 would be performed.

A software test of this type could be written, and as will be discussed, such a software test would be difficult for a subverter to circumvent. Nevertheless, the drawbacks of this software test are severe. Reentrant code is required so that the software test can know (1) the location of the instruction stream, and (2) that no instructions are hidden in data areas. Requiring reentrant programs would in turn require minor changes to the ADEPT–50 Jovial compiler and major changes to the F-level Assembler. A small microprogram change would even be required, so that software could sense the difference between a fetch-protect interrupt and an execute-protect interrupt. Changes would be required to the ADEPT–50 SERVIS, INTRUP, DEBUG, and SKED modules. Were such a software test implemented, run-time overhead would likely be rather high for frequent breakpoint-insertions, since each breakpoint inserted would require execution of 50 or more instructions at run time. Lastly, programmers might not always like having to write reentrant code.

These implementation problems, though described for one specific test approach, are likely to extend to [illegible] to interrupt running user programs via software, since any such test must isolate and augment every user's instruction stream.

Although the above does not demonstrate that detecting user controllable subversion hardware is impossible, it should indicate that by no means is such detection trivial. It is even harder than the above may suggest, for, as the next section indicates, while the subverter is busily building his controllable changes into the CPU he may add insult to injury by inactivating anti-failure and anti-subversion hardware.

Defeat of countermeasures

Hardware countermeasures that are easy to install are easy to remove. Redundant logic can be disabled with the same wire cutters or wire-unwrapper as other logic. This is also true of state-testing and parity-check logic. Alternatively, a clever subverter may add logic that picks up a test signal and uses it to inactivate his own logic while the test is performed. The subverter may, alternatively, arrange things so that tests always pass during his quantum.

Microprogrammed countermeasures are somewhat more difficult for the subverter to inactivate, since he must decode particular ROS addresses to know that a microprogrammed test routine has begun. Once he has done this, though, he may proceed as if he were dealing with hardware, since on a short-term basis, microprogram contents and addresses do not change. The net result of putting countermeasures in ROS rather than in logic is to raise the "work function" for subversion; that is, the subverter must do more work to achieve his goal, but he can reach it.

On the other hand, a thorough software test, such as the breakpoint test previously described, would be nearly impossible to defeat by hardware techniques. This is because of the transient nature of software: a subverter would have to decode program variables such as addresses via hardware; then even minor changes in the breakpoint-placing routines would make that hardware useless. One must recall, however, the large implementation and overhead problems inherent in a user-interrupting software test. In summary, countermeasures can be devised which have a high "work function," but they entail major costs in implementation and system efficiency.

Two assumptions have been inherent in this discussion; namely, that the subverter has both knowledge of system hardware (including subversion countermeasures) and means of changing the hardware. This need not be the case, but whether it is depends on administrative rather than technical considerations. Administrative considerations are the next subject.

Administrative policy

Special handling of hardware documentation and engineering changes may be worthwhile when commercial lines of computers are used for secure time-sharing. First, if hardware or microprograms have been added to the computer to test for failures and subversion attempts, the details of the tests should not be obtainable from the computer manufacturer's worldwide network of sales representatives. The fact that testing is done and the technical details of that testing would seem to be legitimate security objects, since a subverter can neutralize testing only if he knows of it. Classification of those documents which relate to testing is a policy question which should be considered. Likewise, redundant hardware, such as a second copy of the PSW bit 15 logic, might be included in the same category.

The second area is that of change control. Presumably the "Customer Engineer" (CE) personnel who perform engineering changes have clearances allowing them access to the hardware, but what about the technical documents which tell them what to do? A clever subverter could easily alter an engineering change wire list to include his modifications, or could send spurious change documentation. A CE would then unwittingly install the subverter's "engineering change." Since it is asking too much to expect a CE to understand on a wire-by-wire basis each change he performs, some new step is necessary if one wants to be sure that engineering changes are made for technical reasons only. In other words, the computer manufacturer's engineering changes are security objects in the sense that their integrity must be guaranteed. Special paths of transmittal and post-installation verification by the manufacturer might be an adequate way to secure engineering changes; there are undoubtedly other ways. It is clear that a problem exists.

Finally, it should be noted that the 360/50 ROS storage elements, or any equivalent parts of another manufacturer's hardware that contain all system microprogramming, ought to be treated in a special manner, such as physically sealing them in place as part of hardware certification. New storage elements containing engineering changes are security objects of even higher order than regular engineering-change documents, and should be handled accordingly, from their manufacture through their installation.

GENERALIZATIONS AND CONCLUSIONS

Some general points about hardware design that relate to secure time-sharing and some short-range and

Fail-secure vs. fail-soft hardware

Television programs, novels, and motion pictures have made it well known that if something is "fail-safe," it doesn't blow up when it fails. In the same vein, designers of high-reliability computers coined the term "fail-soft" to describe a machine that degrades its performance when a failure occurs, instead of becoming completely useless. It is now proposed to add another term to this family: "Fail-secure: to protect secure information regardless of failure."

The ability to detect failures is a prerequisite for fail-secure operation. However, all system provisions for corrective action based on failure detection must be carefully designed, particularly when hardware failure correction is involved. Two cases were recently described wherein a conflict arose between hardware and software that had been included to circumvent failures.* Automatic correction hardware could likewise mask problems which should be brought to the attention of the System Security Officer via security software.

Clearly, something between the extremes of system crash and silent automatic correction should occur when hardware fails. Definition of what *does* happen upon failure of critical hardware should be a design requirement for fail-secure time-sharing systems. Fail-soft computers are not likely to be fail-secure computers, nor vice versa unless software and hardware have been designed with both concepts in mind.

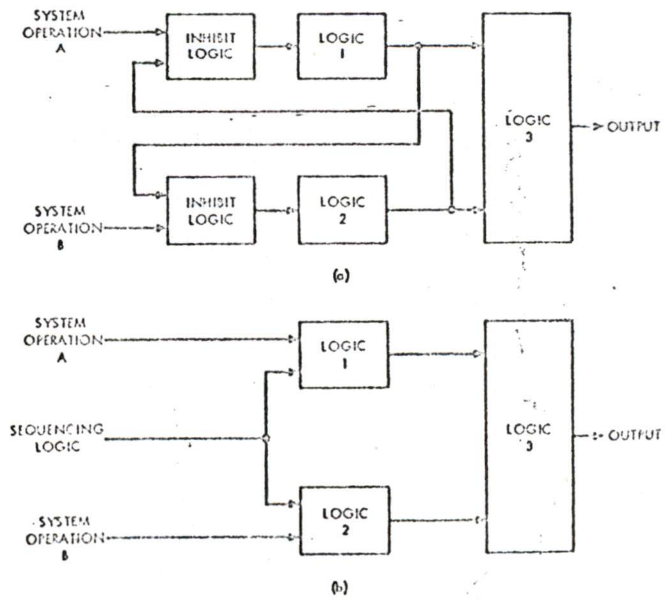
Failure detection by faulty system operation

Computer hardware logic can be grouped by the system operation or operations it helps perform. Some logic—for example, the clock distribution logic—helps perform only one system operation. Other logic—such as the read-only storage address logic in the 360/50—helps perform many system operations, from floating point multiplication to memory protection interrupt handling. When logic is needed by more than one system operation, it is cross-checked for proper performance: Should an element needed for system operations A and B fail, the failure of system operation B would indicate the malfunction of this portion of operation A's logic.

Such interdependence is quite useful in a fail-secure system, as it allows failures to be detected by faulty system operation—a seemingly inelegant error detection mechanism, yet one which requires neither software nor hardware overhead. Some ideas on its uses and limitations follow.

The result of a hardware logic failure can usually be defined in terms of what happens to the system operations associated with the dead hardware. Some logic failure modes are detectable, because they make logic elements downstream misperform unrelated system operations. Analysis will also reveal failure modes which spoil only the system operation which they help perform. These failures must be detected in some other way. There are also, but more rarely, cases where a hardware failure may lead to an operation failure that is not obvious. In the 360/50, a failure could cause skipping of a segment of a control microprogram that wasn't really needed on that cycle. Such failures are not detectable by faulty system operation at least part of the time.

Advantage may be taken of this failure-detection technique in certifying hardware to be fail-secure as well as in original hardware design. In general, the more interdependencies existing among chunks of logic, the more likely are failures to produce faulty system operation. For example, in many places in a computer one finds situations as sketched in Figure 1. Therein,



*At the "Workshop on Hardware-Software Interaction for System Reliability and Recovery in Fault-Tolerant Computers," held July 14–15, 1969 at Pacific Palisades, California, J. W. Herndon of Bell Telephone Labs reported that a problem had arisen in a developmental version of Bell's "Electronic Switching System." It seems that an elaborate setup of relays would begin reconfiguring a bad communications channel at the same time that software in ESS was trying to find out what was wrong. R. F. Thomas, Jr. of the Los Alamos Scientific Laboratory, having had a similar problem with a self-checking data acquisition system, agreed with Herndon that hardware is not clever enough to know what to do about system failures; software [illegible]

TABLE 1—Control Signal Error Detection by Odd Parity Check on Odd-Length Data Field

DATA BITS 012 P	*MEANING*
000 0	data error or control logic error*
000 1	0
001 0	1
001 1	data error
010 0	2
010 1	data error
011 0	data error
011 1	3
100 0	4
100 1	data error
101 0	data error
101 1	5
110 0	data error
110 1	6
111 0	7
111 1	data error or control logic error**

*Control logic incorrectly set all bits to zero.
**Control logic incorrectly set all bits to one.

System Operation A needs the services of Logic Group 1 and Logic Group 3, while System Operation B needs Logic Group 2 and Logic Group 3. Note at this point that, as above, if System Operation A doesn't work because of a failure in Logic Group 3, we have concurrently detected a failure in the logic supporting System Operation B.

A further point is made in Figure 1. Often System Operations A and B must be mutually exclusive; hardware must be added to prevent simultaneous activation of A and B. Two basic design approaches may be taken to solve this problem. An "inhibiting" scheme may be used, wherein logic is added that inhibits Logic Group 1 when Logic Group 2 is active, and vice versa. This approach is illustrated by Figure 1(a). Alternatively, a "sequencing" scheme may be used, wherein logic not directly involved with 1 or 2—such as system clock, mode selection logic, or a status register—defines when A and B are to be active. This approach is illustrated by Figure 1(b).

Now, "inhibit" logic belongs to a particular System Operation, for its function is to asynchronously, on demand, condition the hardware to perform that System Operation. It depends on nothing else; if it fails by going permanently inactive, only its System Operation is affected, and no alarm is given. On the other hand, "sequencing" logic feeds many areas of the machine; its failure is highly likely to be detected by faulty system operation.

A further point can be made here which may be somewhat controversial: that an overabundance of "inhibit"-type asynchronous logic is a good indicator of sloppy design or bad design coordination. While a certain amount must exist to deal with asynchronous pieces of hardware, often it is put in to "patch" problems that no one realized were there till system checkout time. Evidence of such design may suggest more thorough scrutiny is desirable.

System Operations can be grouped by their frequency of occurrence: some operations are needed every CPU cycle, some when the programmer requests them, some only during maintenance, and so on. Thus, some logic which appears to provide a cross-check on other logic may not do so frequently or predictably enough to satisfy certification requirements.

To sum up, the fact that a system crashes when a hardware failure occurs, rather than "failing soft" by continuing to run without the dead hardware, may be a blessing in disguise. If fail-soft operation encompasses hardware that is needed for continued security, such as the memory protection hardware, fail-soft operation is not fail-secure.

Data checking and control signal errors

Control signals which direct data transfers will often be checked by logic that was put in only to verify data purity. The nature and extent of this checking is dependent on the error-detection code used and upon the length of the data field (excluding check bits).

What happens is that if logic fails which controls a data path and its check bits, the data will be forced to either all zeros or all ones. If one or both of these cases is illegal, the control logic error will be detected when the data is checked. (Extensive parity checking on the 360/50 CPU results in much control logic failure detection capability therein.) Table 1 demonstrates an example of this effect; Table 2 describes the conditions for which it exists for the common parity check.

TABLE 2—Control Signal Error Detection by Parity Checking

DATA FIELD LENGTH:	PARITY:	CONTROL LOGIC ERROR CAUSES: all zeros	all ones
even	odd	CAUGHT	MISSED
even	even	MISSED	CAUGHT
odd	odd	CAUGHT	CAUGHT
odd	even	MISSED	[illegible]

CONCLUSIONS

From a short-range viewpoint, 360/50 CPU hardware has some weak spots in it but no holes, as far as secure time-sharing is concerned. Furthermore, the weak spots can be reinforced with little expense. Several alternatives in this regard have been described.

From a longer-range viewpoint, anyone who contemplates specifying a requirement for hardware certification should know what such an effort involves. As reference, some notes are appropriate as to what it took to examine the 360/50 memory protection system to the level required for meaningful hardware certification. The writer first obtained several publications which describe the system. Having read these, the writer obtained the logic diagrams, went to the beginning points of several operations, and traced logic forward. Signals entering a point were traced backward until logic was found which would definitely cause faulty machine operation outside the protection system if it failed. During this tedious process, discrepancies arose between what had been read and what the logic diagrams appeared to show. Some discrepancies were resolved by further study; some were accounted for by special features on the SDC 360/50; some remain.

After logic tracing, the entire protection system was sketched out on eight $8\frac{1}{2} \times 11$ pages. This drawing proved to be extremely valuable for improving the writer's understanding, and enabled failure-mode charting that would have been intractable by manual means from the manufacturer's logic diagrams.

For certifying hardware, documentation quality and currentness is certainly a problem. The manufacturer's publications alone are necessary but definitely not sufficient, because of version differences, errors, oversimplifications, and insufficient detail. Both these and machine logic diagrams are needed.

Though the hardware certification outlook is bleak, an alternative does exist: testing. As previously described, it is possible to require inclusion of low-overhead functional testing of critical hardware in a secure computing system. The testing techniques, whether embedded in hardware, microprograms, or software, could be put under security control if some protection against hardware subversion is desired. Furthermore, administrative security control procedures should extend to "Customer Engineer" activity and to engineering change documentation to the extent necessary to insure that hardware changes are made for technical reasons only.

Careful control of access to computer-based information is, and ought to be, of general concern today. Access controls in a secure time-sharing system such as ADEPT-50 are based on hardware features.[7] The latter deserve scrutiny.

REFERENCES

1 L MOLHO
Hardware reliability study
SDC N-(L)-24276/126/00 December 1969

2 R LINDE C WEISSMAN C FOX
The ADEPT-50 time-sharing system
Proceedings of the Fall Joint Computer Conference Vol 35 p 39-50 1969
Also issued as SDC document SP-3344

3 W H WARE
Security and privacy in computer systems
Proceedings of the Spring Joint Computer Conference Vol 30 p 279-282 1967

4 W H WARE
Security and privacy: Similarities and differences
Proceedings of the Spring Joint Computer Conference Vol 30 p 287-290 1967

5 S G TUCKER
Microprogram control for system/360
IBM Systems Journal Vol 6 No 4 p 222-241 1967

6 G C VANDLING D E WALDECKER
The microprogram control technique for digital logic design
Computer Design Vol 8 No 8 p 44-51 August 1969

7 C WEISSMAN
Security controls in the ADEPT-50 time-sharing system
Proceedings of the Fall Joint Computer Conference Vol 35 p 119-133 1969
Also issued as SDC document SP-3342

Security and privacy: similarities and differences

by WILLIS H. WARE
The RAND Corporation
Santa Monica, California

For the purposes of this paper we will use the term "security" when speaking about computer systems which handle classified defense information, and "privacy" in regard to those computer systems which handle non-defense information which nonetheless must be protected because it is in some respect sensitive. It should be noted at the outset that the context in which security must be considered is quite different from that which can be applied to the privacy question. With respect to classified military information there are federal regulations which establish authority, and discipline to govern the conduct of people who work with such information. Moreover, there is an established set of categories into which information is classified. Once information is classified Confidential, Secret, or Top Secret, there are well-defined requirements for its protection, for controlling access to it, and for transmitting it from place to place. In the privacy situation, analogous/conditions may exist only in part or not at all.

There are indeed Federal and State statutes which protect the so-called "secrecy of communication." But it remains to be established that these laws can be extended to cover or interpreted as applicable to the unauthorized acquisition of information from computer equipment. There are also laws against thievery; and at least one case involving a programmer and theft of privileged information has been tried. The telephone companies have formulated regulations governing the conduct of employees (who are subject to "secrecy of communication" laws) who may intrude on the privacy of individuals; perhaps this experience can be drawn upon by the computer field.

Though there apparently exist fragments of law and some precedents bearing on the protection of information, nonetheless the privacy situation is not so neatly circumscribed and tidy as the security situation. Privacy simply is not so tightly controlled. Within computer networks serving many companies, organizations, or agencies, there may be no uniform governing authority; an incomplete legal framework; no established discipline, or perhaps not even a code of ethics among users. At present there is not even a commonly accepted set of categories to describe levels of sensitivity for private information.

Great quantities of private information are being accumulated in computer files; and the incentives to penetrate the safeguards to privacy are bound to increase. Existing laws may prove inadequate, or may need more vigorous enforcement. There may be need for a monitoring and enforcement establishment analogous to that in the security situation. In any event, it can not be taken for granted that there now exist adequate legal and ethical umbrellas for the protection of private information.

The privacy problem is really a spectrum of problems. At one end, it may be necessary to provide only a very low level of protection to the information for only a very short time; at the opposite end, it may be necessary to invoke the most sophisticated techniques to guarantee protection of information for extended periods of time. Federal regulations state explicitly what aspect of national defense will be compromised by unauthorized divulgence of each category of classified information. There is no corresponding particularization of the privacy situation; the potential damage from revealing private information is nowhere described in such absolute terms. It may be that a small volume of information leaked from a private file may involve inconsequential risk. For example, the individual names of a company's employees is probably not even sensitive, whereas the complete file of employees could well be restricted. Certainly the "big brother" spectre raised by recent Congressional hearings on "invasion of privacy" via massive computer files is strongly related to the volume of information at risk.

Because of the diverse spread in the privacy situation, the appearance of the problem may be quite different from its reality. One would argue on principle that maximum protection should be given to all information labeled private; but if privacy of information is not protected by law and authority, we can expect that the owner of sensitive information will require a system designed to guarantee protection only against the threat as he sees it. Thus, while we might imagine very sophisticated attacks against private files, the reality of the situation may be that much simpler levels of protection will be accepted by the owners of the information.

In the end, an engineering trade-off question must be assessed. The value of private information to an outsider will determine the resources he is willing to expend to acquire it. In turn, the value of the information to its owner is related to what he is willing to pay to protect it. Perhaps this game-like situation can be played out to arrive at a rational basis for establishing the level of protection. Perhaps a company or governmental agency—or a group of companies or agencies, or the operating agent of a multi-access computer service—will have to establish its own set of regulations for handling private information. Further, a company or agency may have to establish penalties for infractions of these regulations, and perhaps even provide extra remuneration for those assuming the extraordinary responsibility of protecting private information.

The security measures deemed necessary for a multi-processing remote terminal computer system operating in a military classified environment have been discussed in the volume.* This paper will compare the security situation with the privacy situation, and suggest issues to be considered when designing a computer system for guarding private information. Technology which can be applied against the design problem is described elsewhere.†

First of all, note that the privacy problem is to some extent present whenever and wherever sharing of the structures of a computer system takes place. A time-sharing system slices time in such a way that each user gets a small amount of attention on some periodic basis. More than one user program is resident in the central storage at one time; and hence, there are obvious opportunities for leakage of information from one program to another, although the problem is alleviated to some extent in systems operating in an interpretive software mode. In a multi-programmed computer system it is also true that more than one user program is normally resident in the core store at a time. Usually, a given program is not executed without interruption; it must share the central storage and perhaps other levels of storage with other programs. Even in the traditional batch-operated system there can be a privacy problem. Although only one program is usually resident in storage at a time, parts of other programs reside on magnetic tape or discs; in principle, the currently executing program might accidentally reference others, or cause parts of previous programs contained on partially re-used magnetic tape to be outputed.

Thus, unless a computer system is completely stripped of other programs—and this means clearing or removing access to all levels of storage—privacy infractions are possible and might permit divulgence of information from one program to another.

Let us now reconsider the points raised in the Peters* paper and extend the discussion to include the privacy situation.

(1) The problem of controlling user access to the resource-sharing computer system is similar in both the security and privacy situations. It has been suggested that one-time passwords are necessary to satisfactorily identify and authenticate the user in the security situation. In some university time-sharing systems, permanently assigned passwords are considered acceptable for user identification. Even though printing of a password at the console can be suppressed, it is easy to ascertain such a password by covert means; hence, repeatedly used passwords may prove unwise for the privacy situation.

(2) The incentive to penetrate the system is present in both the security and privacy circumstances. Revelation of military information can degrade the country's defense capabilities. Likewise, divulgence of sensitive information can to some extent damage other parties or organizations. Private information will always have some value to an outside party, and it must be expected that penetrations will be attempted against computer systems handling such information. It is conceivable that the legal liability for unauthorized leaking of sensitive information may become as severe as for divulging classified material.

(3) The computer hardware requirements appear to be the same for the privacy and security situations. Such features as memory read-write protection, bounds registers, privileged instructions, and a privileged mode of operation are required to protect

*Peters, B., "Security Considerations in a Multi-Programmed System".

†Petersen, H. E., and R. Turn, Systems Implications of Privacy."

*Peters, B., *loc cit.*

information, be it classified or sensitive. Also, overall software requirements seem similar, although certain details may differ in the privacy situation because of communication matters or difference in user discipline.

(4) The file access and protection problem is similar under both circumstances. Not all users of a shared computer-private system will be authorized access to all files in the system, just as not all users of a secure computer system will be authorized access to all files. Hence, there must be some combination of hardware and software features which controls access to the on-line classified files in conformance with security levels and need-to-know restrictions and in conformance with corresponding attributes in the privacy situation. As mentioned earlier, there may be a minor difference relative to volume. In classified files, denial of access must be absolute, whereas in private files access to a small quantity of sensitive information might be an acceptable risk.

(5) The philosophy of the overall system organization will probably have to be different in the privacy situation. In the classified defense environment, users are indoctrinated in security measures and their personal responsibility can be considered as part of the system design. Just as the individual who finds a classified document in a hallway is expected to return it, so the man who accidentally receives classified information at his console is expected to report it. The users in a classified system are subject to the regulations, authority, and discipline of a governmental agency. Similar restrictions may not prevail in a commercial or industrial resource-sharing computer network, nor in government agencies that do not operate within the framework of government classification. In general, it would appear that one cannot exploit the good will of users as part of a privacy system's design. On the other hand, the co-operation of users may be part of the design philosophy if it proves possible to impose a uniform code of ethics, authority, and discipline within a multi-access system. Uniform rules of behavior might be possible if all users are members of the same organization, but quite difficult or impossible if the users are from many companies or agencies.

(6) The certifying authority is certainly different in the two situations. It is easy to demonstrate that the total number of internal states of a computer is so enormous that some of them will never prevail in the lifetime of the machine. It is equally easy to demonstrate that large computer programs have a large number of internal paths, which implies the potential existence of error conditions which may ap-

governing the internal scheduling and operation of multi-programmed, time-sharing or batch-operated machines are likely to be extensive and complex; and if security or privacy is to be guaranteed, some authority must certify that the monitor is properly programmed and checked out. Similarly, the hardware must also be certified to possess appropriate protective devices.

In a security situation, a security officer is responsible for establishing and implementing measures for the control of classified information. Granted that he may have to take the word of computer experts or become a computer expert himself, and granted that of itself his presence does not solve the computer security problem, there is nonetheless at least an assigned, identifiable responsible authority. In the case of the commercial or industrial system, who is the authority? Must the businessman take the word of the computer manufacturer who supplied the software? If so, how does he assure himself that the manufacturer hasn't provided "ins" to the system that only he, the manufacturer, knows about? Must the businessman create his own analog of defense security practices?

(7) Privacy and security situations are certainly similar in that deliberate penetrations must be anticipated, if not expected; but industrial espionage against computers may be less serious. On the other hand, industrial penetrations against computers could be very profitable and perhaps safer from a legal viewpoint.

It would probably be difficult for a potential penetrator to mount the magnitude of effort against an industrial resource-sharing computer system that foreign agents are presumed to mount against secrecy systems of other governments. To protect against large-scale efforts, an industry-established agency could keep track of major computing installations and know where penetration efforts requiring heavy computer support might originate. On the other hand, the resourceful and insightful individual can be as great a threat to the privacy of a system. If one can estimate the nature and extent of the penetration effort expected against an industrial system, perhaps it can be used as a design parameter to establish the level of protection for sensitive information.

(8) The security and privacy situations are certainly similar in that each demands secure communication circuits. For the most part, methods for assuring the security of communication channels have been the exclusive domain of the military and government. What about the non-government user? Could the specifications levied on common carriers in their

implied warranty of a private circuit be extended? Does the problem become one for the common carriers? Must they develop communication security equipment? If the problem is left to the users, does each do as he pleases? Might it be feasible to use the central computer itself to encode information prior to transmission? If so, the console will require special equipment for decoding the messages.

(9) Levels of protection for communications are possibly different in the two situations. If one believes that a massive effort at penetration could not be mounted against a commercial private network, a relatively low-quality protection for communication would be sufficient. On the other hand, computer networks will inevitably go international. Then what? A foreign industry might find it advantageous to tap the traffic of U.S. companies operating an international and presumably private computer network. Might it be that for reasons of national interest we will someday find the professional cryptoanalytic effort of a foreign government focused on the privacy-protecting measures of a computer network?

If control of international trade were to become an important instrument of government policy, then any international communications network involved with industrial or commercial computer-private systems will need the best protection that can be provided.

This paper has attempted to identify and briefly discuss the differences and similarities between computer systems operating with classified military information and computer systems handling private or sensitive information. Similar hardware and software and systems precautions must be taken. In most respects, the differences between the two situations are only of degree. However, there are a few aspects in which the two situations genuinely differ in kind, and on these points designers of a system must take special note. The essential differences between the two situations appear to be the following:

(1) Legal foundations for protecting classified information are well established, whereas in the privacy situation a uniform authority over users and a penalty structure for infractions are lacking. We may not be able to count on the good will and disciplined behavior of users as part of the protective measures.

(2) While penetrations can be expected against both classified and sensitive information, the worth of the material at risk in the two situations can be quite different, not only to the owner of the data but also to other parties and to society.

(3) The magnitude of the resources available for protection and for penetration are markedly smaller in the privacy situation.

(4) While secure communications are required in both situations, there are significant differences in details. In the defense environment, protected communications are the responsibility of a government agency, appropriate equipment is available, and the importance of protection over-rides economic considerations. In the privacy circumstance, secure satisfactory communication equipment is generally not available, and the economics of protecting communications is likely to be more carefully assessed.

(5) Some software details have to be handled differently in the privacy situation to accommodate differences in the security of communications.

It must be remembered that since the Federal authority and regulations for handling classified military information do not function for private or sensitive information, it does not automatically follow that a computer network designed to safely protect classified information will equally well protect sensitive information. The all important difference is that the users of a computer-private network may not be subject to a common authority and discipline. But even if they are, the strength of the authority may not be adequate to deter deliberate attempts at penetration.

Chairman's Introduction to the SJCC Session

Security and privacy in computer systems

by WILLIS H. WARE
The RAND Corporation
Santa Monica, California

INTRODUCTION

Information leakage in a resource-sharing computer system

With the advent of computer systems which share the resources of the configuration among several users or several problems, there is the risk that information from one user (or computer program) will be coupled to another user (or program). In many cases, the information in question will bear a military classification or be sensitive for some reason, and safeguards must be provided to guard against the leakage of information. This session is concerned with accidents or deliberate attempts which divulge computer-resident information to unauthorized parties.

Espionage attempts to obtain military or defense information regularly appear in the news. Computer systems are now widely used in military and defense installations, and deliberate attempts to penetrate such computer systems must be anticipated. There can be no doubt that safeguards must be conceived which will protect the information in such computer systems. There is a corresponding situation in the industrial world. Much business information is company-confidential because it relates to proprietary processes or technology, or to the success, failure, or state-of-health of the company. One can imagine a circumstance in which it would be profitable for one company to mount an industrial espionage attack against the computer system of a competitor. Similarly, one can imagine scenarios in which confidential information on individuals which is kept within a computer is potentially profitable to a party not authorized to have the information. Hence, we can expect that penetrations will be attempted against computer systems which contain non-military information.

This session will not debate the existence of espionage attempts against resource-sharing systems. Rather, it is assumed that the problem exists, at least in principle if not in fact, and our papers will be devoted to discussing technological aspects of the problem and possible approaches to safeguards.

First of all, clarification of terminology is in order. For the military or defense situation, the jargon is well established. We speak of "classified information," "military security," and "secure computer installations." There are rules and regulations governing the use and divulgence of military-classified information, and we need not dwell further on the issue. In the non-military area, terminology is not established. The phrase "industrial security" includes such things as protecting proprietary designs and business information; but it also covers the physical protection of plants and facilities. For our purposes, the term is too broad. In most circles, the problem which will concern us is being called the "privacy problem."

The words "private" and "privacy" are normally associated with an individual in a personal sense, but *Webster's Third New International Dictionary* also provides the following definitions:

Private:...	intended for or restricted to the use of a particular person, or group, or class of persons; not freely available to the public.
Privacy:...	isolation, seclusion, or freedom from unauthorized oversight or observation.

We are talking about restricting information within a computer for the use of a specified group of persons; we do not want the information freely available to the public. We want to isolate the information from unauthorized observation. Hence, the terminology appears appropriate enough, although one might hope that new terms will be found that do not already have strongly established connotations. For our purposes today, "security" and "classified"

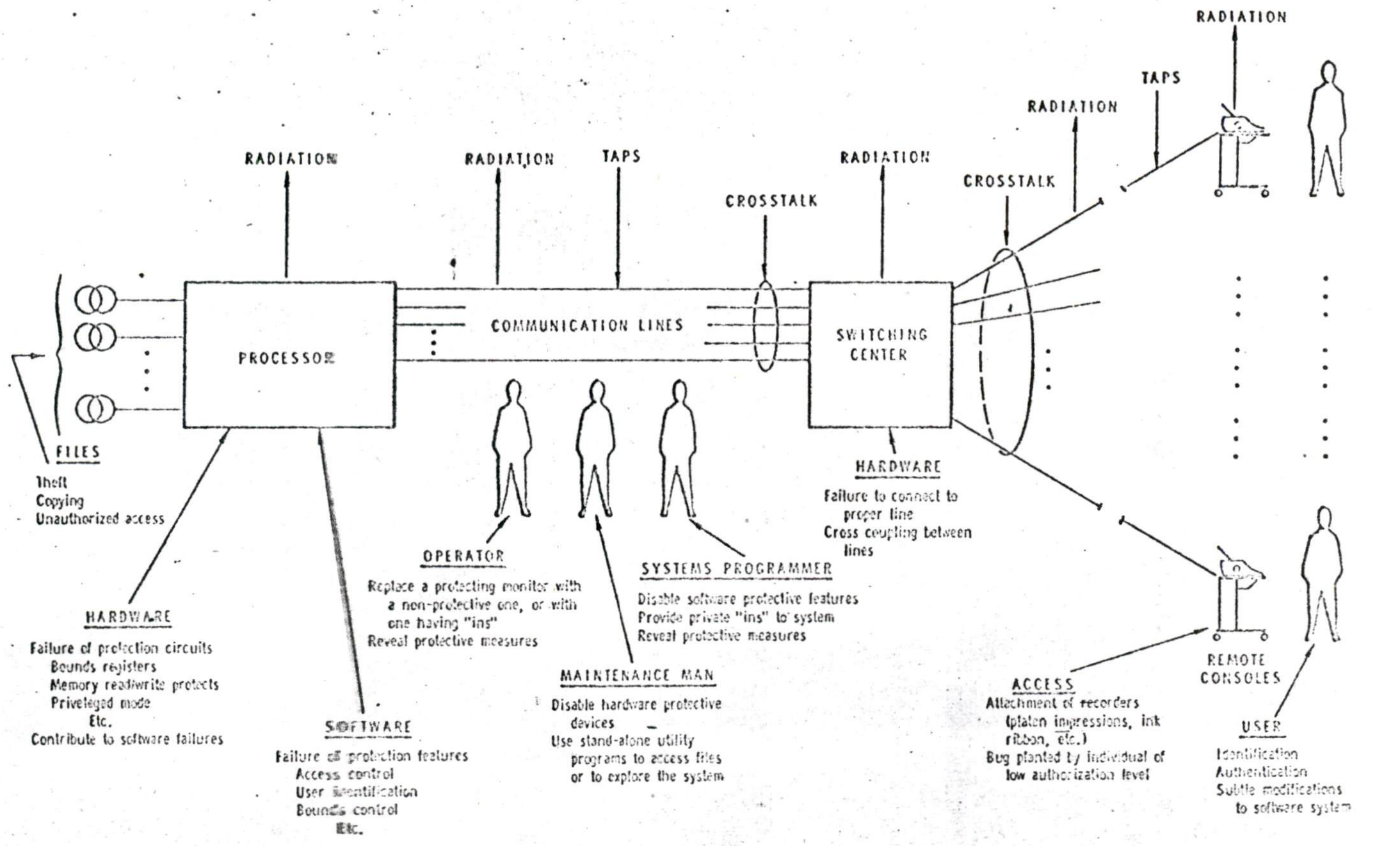

RADIATION
RADIATION
TAPS
CROSSTALK
RADIATION
CROSSTALK
RADIATION
TAPS
RADIATION
PROCESSOR
COMMUNICATION LINES
SWITCHING CENTER
FILES
Theft
Copying
Unauthorized access
HARDWARE
Failure of protection circuits
Bounds registers
Memory read/write protects
Priveleged mode
Etc.
Contribute to software failures
SOFTWARE
Failure of protection features
Access control
User identification
Bounds control
Etc.
OPERATOR
Replace a protecting monitor with a non-protective one, or with one having "ins"
Reveal protective measures
MAINTENANCE MAN
Disable hardware protective devices
Use stand-alone utility programs to access files or to explore the system
SYSTEMS PROGRAMMER
Disable software protective features
Provide private "ins" to system
Reveal protective measures
HARDWARE
Failure to connect to proper line
Cross coupling between lines
ACCESS
Attachment of recorders (platen impressions, ink ribbon, etc.)
Bug planted by individual of low authorization level
REMOTE CONSOLES
USER
Identification
Authentication
Subtle modifications to software system

Figure 1 – Typical configuration of resource-sharing computer system

will refer to military or defense information or situations; "private" or "privacy," to the corresponding industrial, or non-military governmental situations. In each case, the individual authorized to receive the information will have "need to know" or "access authorization."

We will do the following in this session. In order to bring all of us to a common level of perspective on resource-sharing computer systems, I will briefly review the configuration of such systems and identify the major vulnerabilities to penetration and to leakage of information. The following paper by Mr. Peters will describe the security safeguards provided for a multi-programmed remote-access computer system. Then I will contrast the security and privacy situations, identifying similarities and differences. The final paper by Dr. Petersen and Dr. Turn will discuss technical aspects of security and privacy safeguards. Finally, we have a panel of three individuals who have faced the privacy problem in real-life systems; each will describe his views toward the problem, and his approach to a solution. In the end, it will fall upon each of you to conceive and implement satisfactory safeguards for the situation which concerns you.

A priori, we cannot be certain how dangerous a given vulnerability might be. Things which are serious for some computer systems may be only a nuisance for others. Let us take the point of view that we will not prejudge the risk associated with a given vulnerability or threat to privacy. Rather, let us try only to suggest some of the ways in which a computer system might divulge information to an unauthorized party in either the security or the privacy situation. We'll leave for discussion in the context of particular installations the question of how much protection we want to provide, what explicit safeguards must be provided, and how serious any particular vulnerability might be.

The hardware configuration of a typical resource-sharing computer system is shown in Figure 1. There is a central processor to which are attached computer-based files and a communication network for linking to remote users via a switching center. We observe first of all that the files may contain information of different levels of sensitivity or military classification; therefore, access to these files by users must be controlled. Improper or unauthorized access to a file can divulge information to the wrong person. Certainly, the file can also be stolen—a rather drastic divulgence of information. On the other hand, an unauthorized copy of a file might be made using the computer itself, and the copy revealed to unauthorized persons.

The central processor has both hardware and software components. So far as hardware is concerned, the circuits for such protections as bound registers, memory read-write protect, or privileged mode might fail and permit information to leak to improper destinations. A large variety of hardware failures might contribute to software failures which, in turn, lead to divulgence. Since the processor consists of high-speed electronic circuits, it can be expected that large quantities of electromagnetic energy will radiate; conceivably an eavesdropping third party might acquire sensitive information. Failure of the software may disable such protection features as access control, user identification, or memory bounds control, leading to improper routing of information.

Intimately involved with the central computer are three types of personnel: operators, programmers, and maintenance engineers. The operator who is responsible for minute-by-minute functioning of the system might reveal information by doing such things as replacing the correct monitor with a non-protecting one of his own, or perhaps with a rigged monitor which has special "ins" for unauthorized parties. Also, he might reveal to unauthorized parties some of the protective measures which are designed into the system. A co-operative effort between a clever programmer and an engineer could "bug" a machine for their own gain in such a sophisticated manner that it might remain unnoticed for an extended period. ("Bug" as just used does not refer to an error in a program, but to some computer equivalent of the famous transmitter in a martini olive.) Bugging of a machine could very easily appear innocent and open.

Operator-less machine systems are practical, and in principle one might conjecture that a machine could be bugged by an apparently casual passerby. There are subtle risks associated with the maintenance process. While attempting to diagnose a system failure, information could easily be generated which would reveal to the maintenance man how the software protections are coded. From that point, it might be easy to rewire the machine so that certain instructions appeared to behave normally, whereas in fact, the protective mechanisms could be bypassed.

While some of the things that I've just proposed require deliberate acts, others could happen by accident.

Thus, so far as the computing central itself is concerned, we have potential vulnerabilities in control of access to files; in radiation from the hardware; in hardware, software, or combined hardware-soft-

The communication links from the central processor to the switching center, and from the switching center to the remote consoles are similarly vulnerable. Any of the usual wiretapping methods might be employed to steal information from the lines. Since some communications will involve relatively high-frequency signals, electromagnetic radiation might be intercepted by an eavesdropper. Also, crosstalk between communication links might possibly reveal information to unauthorized individuals. Furthermore, the switching central itself might have a radiation or crosstalk vulnerability; it might fail to make the right connection and so link the machine to an incorrect user.

A remote console might also have a radiation vulnerability. Moreover, there is the possibility that recording devices of various kinds might be attached to the console to pirate information. Consideration might have to be given to destroying the ribbon in the printing mechanism, or designing the platen so that impressions could not be read from it.

Finally, there is the user of the system. Since his link to the computer is via a switching center, the central processor must make certain with whom it is conversing. Thus, there must be means for properly identifying the user; and this means must be proof against recording devices, pirating, unauthorized use, et . Even after a user has satisfactorily established his identity, there remains the problem of verifying his right to have access to certain files, and possibly to certain components of the configuration. There must be a means for authenticating the requests which he will make of the system, and this means must be proof against bugging, recorders, pirating, unauthorized usage, etc. Finally, there is the ingenious user who skillfully invades the software system sufficiently to ascertain its structure, and to make changes which are not apparent to the operators or to the systems programmers, but which give him "ins" to normally unavailable information.

To summarize, there are human vulnerabilities throughout; individual acts can accidentally or deliberately jeopardize the protection of information in a system. Hardware vulnerabilities are shared among the computer, the communications system, and the consoles. There are software vulnerabilities; and vulnerabilities in the system's organization, e.g., access control, user identification and authentication. How serious any one of these might be depends on the sensitivity of the information being handled, the class of users, the operating environment, and certainly on the skill with which the network has been invasions which have been suggested plus many readily conceivable.

This discussion, although not an exhaustive consideration of all the ways in which a resource-sharing computer system might be either accidentally or deliberately penetrated for the purposes of unauthorized acquisition of information, has attempted to outline some of the major vulnerabilities which exist in modern computing systems. Succeeding papers in this session will address themselves to a more detailed examination of these vulnerabilities and to a discussion of possible solutions.

Security controls in the ADEPT-50 time-sharing system

by C. WEISSMAN

System Development Corporation
Santa Monica, California

"Authority intoxicates/And makes mere sots of magistrates"--*Butler*

FOREWORD

At present, the system described in this paper has not been approved by the Department of Defense for processing classified information. This paper does not represent DOD policy regarding industrial application of time- or resource-sharing of EDP equipment.

INTRODUCTION

Computer-based, resource sharing systems are, and contain things of value; therefore, they should be protected. The valuables are the information data bases, the processes that manipulate them, and the physical plant, equipment, and personnel that form the system plexus. An extensive lore is developing on the subject of system protection.[1,2] Petersen and Turn[3] discuss in considerable detail the substance of protection of non-military information systems in terms of threats and countermeasures. Ware[4,5] contrasts "security" and "privacy" for viewing protection in militarys ystems as well. This paper describes the security controls implemented in the ADEPT-50 time-sharing system[6]—a resource sharing system designed to handle sensitive information in classified government and military facilities.*

Our approach to security control is based on a set theoretic model of access rights. This approach appears natural, since the important objects of security are sets of things—users, terminals, programs, files—and the operators of set theory—membership, intersection, union—are easily programmed for, and quickly performed by, computer. The formal model defines time-sharing security control of user, terminal, job and file security objects in terms of equations of access based upon their security profiles—a triplet of Authority, Category, and Franchise property sets. The correspondence of these properties to government and military Classification, Compartments, and Need-to-Know is demonstrated. Implementation of the model in the ADEPT-50 Time Sharing System is described in detail, as are features that transcend the model including initialization of the security profiles, the LOGIN decision procedure, system integrity checks, security residue control, and security audit trails. Other novel features of ADEPT security control are detailed and include: automatic file classification based upon the cumulative security history of referenced files; the "security umbrella" of the ADEPT job; and once-only passwords. The paper concludes with a recapitulation of the goals of ADEPT security control, approximate costs of implementation and operation of the security controls, and suggested extensions and improvements.

Historically, protection of a sensitive computer facility has been attained by limiting physical access to the computer room and shielding the computer complex

* Development of ADEPT was supported in part by the Advanced Research Projects Agency of the Department of Defense.

from electromagnetic radiation. This "sheltered" approach promotes one-at-a-time, batch usage of the facility. Modern hardware and software technology has moved forward to more powerful and cost/effective time-shared, multi-access, multiprogrammed systems. However, three features of such systems pose a challenge to the sheltered mode of protection: (1) concurrent multiple users with different access rights operating remote from the shielded room; (2) multiple programs with different access rights co-resident in memory; and (3) multiple files of different data sensitivities simultaneously accessible. These features appear to violate traditional methods of accountability based upon a single user (or multiple users with like clearances) operating within strictly controlled facilities. The problem is of such magnitude that no time-sharing system has yet been certified for use in the manner described! However, some multi-access systems are in operation in a classified mode,[7,8] and a number of design approaches have been suggested.[9,10,11,12]

In addition to the usual goal of building an effective time-sharing system,[13] the ADEPT project began with a number of security objectives as well:

1. Build a security control mechanism that supports heterogeneous levels and types of classifications.
2. Design the security control mechanism in such a manner that it is itself unclassified until primed by security configuration parameters, a point strongly supported by Baran[14] regarding communicatons security.
3. Construct the security control mechanism as an isolated portion of the total time-sharing system so that it may be carefully scrutinized for correctness, completeness, and reliability.
4. Do the above in as frugal a manner as possible, considering costs to design, fabricate, and operate. Good system performance is our principal criterion in selecting among alternative technical solutions, as noted by the author elsewhere.[15]

In approaching our task, we recognize security as a total system problem involving hardware, communication, personnel, and software safeguards. However, our focus is primarily on monitor software, and its interfaces with the other areas. This view is not parochial: our hardware is a standard IBM 360 model 50; communication security is an established field of study with considerable technological know-how;[14] and the policy, doctrine, and procedures for personnel behavior in classified environments are extensive, with legal foundations. Thus, our only degree of freedom is the control we build into the time-sharing executive software.

A security control formalism

A formal model of software security control for access to sensitive portions of ADEPT is developed here.

Security objects

Four kinds of security objects are to be managed by our model: user, terminal, job, and file. Let *u* denote some user; *t* some terminal; *j* some job; and *f* some file.

Security properties

Each security object is described by a security profile that is an ordered triplet of security properties—Authority (A), Category (C), and Franchise (F). Authority is a set of hierarchically ordered security jurisdictions. Category is a set of discrete security jurisdictions. Franchise is a set of users licensed with privileged security jurisdiction.

The property "Authority" is defined as a set A, where

$$A = \{a^0 < a^1 <, \cdots, < a^n\} \qquad (1)$$

and the specific members, a^i, of the set are security jurisdictions hierarchically ordered.

"Category" is a discrete set of specific compartments, c^i,

$$C = \{c^0, c^1, \cdots, c^k\} \qquad (2)$$

Compartments are mutually exclusive security sanctuaries with discrete jurisdictions.

"Franchise" is a security jurisdiction privileged to a given set of users, i.e.,

$$F = \{u | u \text{ is a user}\} \qquad (3)$$

For a given terminal, *t*, let a given Authority set, A, be denoted by A_t, or in general, let a given security object, α, denote a given property, P, for α as P_α. Hence we can speak of A_u, or C_j, etc., to mean the specific Authority set for a given user, *u*, or the specific Category set for a given job, *j*, respectively.

Four important sets (of users) arise with respect to the Franchise property, namely, Franchise for files, terminals, jobs, and users. To distinguish the sense in which a given user is being considered, we subscript u by the security object under consideration. Hence, u_f means the user with jurisdiction to file *f*; u_t and u_j are similarly defined. For completeness, we define u_u as empty *u*. We can now define Franchise for each security object.

$$F_u = \{u\} \tag{4}$$

$$F_t = \{u_t^0, u_t^1, \cdots, u_t^\lambda\} \tag{5}$$

$$F_j = \{u_j^0, u_j^1, \cdots, u_j^\mu\} \tag{6}$$

$$F_f = \{u_f^0, u_f^1, \cdots, u_f^\nu\} \tag{7}$$

Equation (4) states that the Franchise for a user is restricted to himself; his jurisdiction is unique, and no other user is so endowed. Equation (5) states that the terminal Franchise is possessed by λ different users who have jurisdiction over the terminal *t*. Likewise, equations (6) and (7) define the job and file Franchise sets.

In security discussions, one hears the familiar phrase, "he needs a higher-level clearance." We can now define "higher level" with our model.

Let α and β be security objects and let ρ be some function such that $\rho(A_\alpha)\epsilon A$.

Then,

$$A_\alpha \geq A_\beta \leftrightarrow \rho(A_\alpha) \geq \rho(A_\beta) \tag{8}$$

$$C_\alpha \geq C_\beta \leftrightarrow C_\alpha \supseteq C_\beta \tag{9}$$

$$F_\alpha \geq F_\beta \leftrightarrow F_\alpha \supseteq F_\beta \tag{10}$$

Equation (8) claims that the Authority of a security object, A_α is at a "higher level" than another security object A_β when the specific authority, a_α is greater than the specific authority, a_β.

It is implicit in equations (1) and (8) that the specific authorities, a^i, must be numerically encoded for the magnitude relationships to hold. Equations (9) and (10) define P_α to be greater than P_β if and only if P_β is a subset of P_α.

Events may alter the membership of property sets. Let P_f^e be the *e*th P_f in a given context.

Define the Authority history, A_h, at the *e*th event as

$$A_h(0) = a_f^0 \tag{11}$$

$$A_h(e) = \max\,(A_h(e-1), \rho(A_f^e)), e > 0 \tag{12}$$

Likewise, define the Category history C_h, at the *e*th event as

$$C_h(0) = \phi \tag{13}$$

$$C_h(e) = C_h(e-1) \cup C_f^e, e > 0 \tag{14}$$

Equations (11) through (14) recursively define two useful sets that accumulate a history of file references as a function of file reference events, *e*. A history of the highest Authority, A_h, is defined by equation (12) as either the previous set, $A_h(e-1)$, or the current set, $\rho(A_f^e)$, whichever is larger in the sense of equation (8). Equation (11) gives the initial condition as some low specific file authority, a_f^0. Equation (14) defines the highest Category history as the union of the previous set, $C_h(e-1)$, and the current set, C_f^e; while equation (13) states that the union is initially the empty set.

Though F_h could be defined in our model, no need is seen at this time for a Franchise history. More will be said about these history sets later.

Property determination

Table I presents in a 3 × 4 matrix a summary of the rules for determining the security profile triplets, P_α. We shall examine these rules here. For the user *u*, A_u and C_u are given constants, and F_u is given by equation (4). For the terminal *t*, A_t and C_t are given constants, and F_t is given by equation (5). Given A_u and A_t, we determine A_j as:

$$A_j = \min\,(A_u, A_t) \tag{15}$$

Likewise, given C_u and C_t, we determine C_j as:

$$C_j = C_u \cap C_t \tag{16}$$

Equation (6) gives F_j to complete the job security profile triplet.

An existing file has its security profile predetermined with A_f and C_f as given constants, and F_f as given by equation (7). However, a new file—one just created—derives its security profile from the job's file access history according to the following:

$$A_f = A_h(e) \tag{17}$$

$$C_f = C_h(e) \tag{18}$$

$$F_f = u_j^i \tag{19}$$

From equations (11) through (14) we see how the Authority and Category histories accumulate as a function of event *e*. These events are the specific times when files are accessed by a job. To maintain security

TABLE I—Security property determination matrix

Object \ Property	Authority A	Category C	Franchise F
User, u	Given Constant	Given Constant	u
Terminal, t	Given Constant	Given Constant	u^i_t
Job, j	$\min(A_u, A_t)$	$C_u \cap C_t$	u^i_j
File, f	*Existing file* Given Constant	*Existing file* Given Constant	u^i_f
	New file $\max(A_h(e-1), \rho(A^e_f)), e > 0$	*New file* $C_h(e-1) \cup C^e_f, e > 0$	u^i_j

integrity, these histories can never exceed (i.e., be greater than) the job security profile. This is specified as,

$$A_h(\infty) \rightarrow A_j \qquad (20)$$

$$C_h(\infty) \rightarrow C_j \qquad (21)$$

For $e = 0$, we see the properties initialized to their simplest form. However, as e gets large, the histories accumulate, but never exceed the upper limit set by the job. $A_h(e)$ and $C_h(e)$ are important new concepts, discussed in further detail later. We speak of them, affectionately, as the security "high-water mark," with analogy to the bath tub ring that marks the highest water level attained.

The Franchise of a new file is always obtained from the Franchise of the job given by equation (6). When $i = \mu = 0$, the job is controlled by the single user u_j who becomes the owner and creator of the file with the sole Franchise for the file.

Access control

Our model is now rich enough to express the equations of access control. We wish to control access by a user to the system, to a terminal, and to a file. Access is granted to the system if and only if

$$u \in U \qquad (22)$$

where U is the set of all sanctioned users known to the system.
Access is granted to a terminal if and only if

$$u \in F_t \qquad (23)$$

If equations (22) and (23) hold, then by definition

$$u = u_t = u_j \qquad (24)$$

Access is granted to a file if and only if

$$P_j \geq P_f \qquad (25)$$

for properties A and C according to equations (8) and (9), and

$$u_j \in F_f \qquad (26)$$

If equations (25) and (26) hold, then access is granted and $A_h(e)$ and $C_h(e)$ are calculated by equations (12) and (14).

Model interpretation

Three different dimensions for restricting access to sensitive information and information processes are possible with the security profile triplet. The generality of this technique has considerable application to public and military systems. For the system of interest, however, the Authority property corresponds to the Top Secret, Secret, etc., levels of government and military security; Category corresponds to the host of special control compartments used to restrict access by project and area; such as those of the Intelligence and Atomic Energy communities; and the Franchise property corresponds to access sanctioned on the basis of need-to-know. With this interpretation, the popular security terms "classification" and "clearance" can be defined by our model in the same dimensions—as a min/max test on the security profile triplet. Classification is attached to a security object to designate the minimum security profile required for access, whereas clearance grants to a security object the maximum security profile it has permission to exercise. Thus, legal access obtains if the clearance is greater than or equal to the classification, i.e., if equation (25) holds.

Another observation on the model is the "job umbrella" concept implied by equations (22) through (26); i.e., the derived clearance of the job (not the clearance of the user) is used as the security control triplet for file access. The job umbrella spreads a homogeneous clearance to normalize access to a heterogeneous assortment of program and data files. This simplifies the problem of control in a multi-level security system. Also note how the job umbrella's high-water mark (equations (11) through (14)) is used to automatically classify new files (equations (17) and (18)); this subject is discussed further below.

A final observation on the model is its application of need-to-know to terminal access, equation (23). This feature allows terminals to be restricted to special people and/or special groups for greater control of personnel interfaces—i.e., systems programmers, computer operators, etc.

Security control implementation

The selection of a set theoretic model of security control was not fortuitous, but a deliberate choice biased toward computational efficiency and ease of implementation. It permits the clean separation and isolation of security control code from the security control data, which enables ADEPT's security mechanisms to be openly discussed and still remain safe—a point advocated by others.[14,16] We achieve this safety by "arming" the system with security control data only once at start-up time by the SYSLOG procedure discussed later. Also, the model improves the credibility of the security system, enhancing its understanding and thereby promoting its certification.

Security objects: Identity and structure

Each security object has a unique identification (ID) within the system such that it can be managed individually. The form of the ID depends upon the security-object type; the syntax of each is given below.

User identification

For generality of definition, each user is uniquely identified by his *user:id*, which must be less than 13 characters with no embedded blanks.

The *user:id* can be any meaningful encoding for the local installation. For example, it can be the individual's Social Security number, his military serial number, his last name (if unique and less than 13 characters), or some local installation man-number convention. The set of all *user:ids* constitutes the universal set, *U*.

Terminal identification

All peripheral devices in ADEPT are identified uniquely by their IBM 360 device addresses. Besides interactive terminals, this includes disc drives, tape drives, line printer, card reader-punch, drums, and 1052 keyboard. Therefore, *terminal:id* must be a two-digit hexadecimal number corresponding to the unit address of the device.

Job identification

ADEPT consists of two parts: the Basic Executive (BASEX), which handles the allocation and scheduling of hardware resources, and the Extended Executive (EXEX), which interfaces user programs with BASEX. ADEPT is designed to operate itself and user programs as a set of 4096-byte pages. BASEX is identified as certain pages that are fixed in main core, whereas EXEX and user programs are identified as sets of pages that move dynamically between main and swap memory. A set of user programs are identified as a job, with page sets for each program (the program map) described in the job's environment area, i.e., the job's "state tables." Every job in ADEPT has an environment area that is swapped with the job. It contains dynamic system bookkeeping information pertinent to the job, including the contents of the machine registers (saved when the job is swapped out), internal file and I/O control tables, a map of all the program's pages on drum, *user:id*, and the job security control parameters. The environment page(s) are memory-protected against reading and writing by user programs, as they are really swappable extensions of the monitor's tables.

The *job:id* is then a transitory internal parameter which changes with each user entrance and exit from the system. The *job:id* is a relative core memory address used by the executive as a major index into central system tables. It is mapped into an external two-digit number that is typed to the user in response to a successful LOGIN.

File identification

ADEPT's file system is quite rich in the variety of file types, file organization, and equipment permitted. There are two file types: temporary and permanent.

Temporary files are transitory "scratch" disc files, which disappear from the system inventory when their parent job exits from the system. They are always placed on resident system volumes, and are private to the program that created them.

Permanent files constitute the majority of files cataloged by the system. Their permanence derives from the fact that they remain inventoried, cataloged, and available even after the job that created or last referenced them is no longer present, and even if they are not being used. Permanent files may be placed by the user on resident system volumes or on demountable private volumes.

There are six file organizations from which a user may select to structure the records of his file: Physical-sequential, S1; non-formatted, S2; index-sequential, S3; partitioned, S4; multiple volume fixed record, S5; and single volume fixed record, S9. Regardless of the organization of the records, ADEPT manages them as a collection, called a file. Thus, security control is at the file level only, unlike more definitive schemes of sub-element control.[8,10–12]

All the control information of a file that describes type, organization, physical storage location, date of creation, and security is distinct from the data records of the file, and is the catalog of the file.

All cataloged ADEPT files are uniquely identified by a four-part name; each part has various options and defaults (system assumptions). This name, the *file:id*, has the following form:

file:id : : = *name, form, user:id, volume:id*

Name is a user-generated character string of up to eight characters with no embedded blanks. It must be unique on a private volume as well as for Public files (described below).

Form is a descriptor of the internal coding of a file. Up to 256 encodings are possible, although only these seven are currently applicable:

1 = binary data
2 = relocatable program
3 = non-relocatable program
4 = card images
5 = catalog
6 = DLO (*Delayed Output*)
7 = line images

User:id corresponds to the owner of the file, i.e., the creator of the file.

Volume:id is the unique file storage device (tape, disc, disc pack, etc.) on which the file resides. For various reasons, including reliability, ADEPT file inventories are distributed across the available storage media, rather than centralized on one particular volume. Thus, all files on a given disc volume are inventoried on that volume.

Security properties: Encoding and structure

Implementation of the security properties in ADEPT is not uniform across the security objects as suggested by our model, particularly the Franchise property. Lack of uniformity, brought about by real-world considerations, is not a liability of the system but a reflection of the simplicity of the model. Extensions to the model are developed here in accordance with that actually implemented in ADEPT.

Authority

Authority is fixed at four levels ($\omega = 3$ for equation (1)) in ADEPT, specifically, UNCLASSIFIED, CONFIDENTIAL, SECRET, and TOP SECRET in accordance with Department of Defense security regulations. The Authority set is encoded as a logical 4-bit item, where positional order is important. Magnitude tests are used extensively, such that the high-order bits imply high Authority in the sense of equation (8).

Category

Category is limited to a maximum of 16 compartments ($\psi \leq 15$ for equation (2)), encoded as a logical 16-bit item. Boolean tests are used exclusively on this datum. The definition of (and bit position correspondence to) specific compartments is an installation option at ADEPT start-up time (see SYSLOG). Typical examples of compartments are EYES ONLY, CRYPTO, RESTRICTED, SENSITIVE, etc.

Franchise

Property Franchise corresponds to the military concept of need-to-know. Essentially, this corresponds to a set of *user:ids*; however, the ADEPT implementation of Franchise is different for each security object:

1. User: All users wishing ADEPT service must be known to the system. This knowledge is imparted by SYSLOG at start-up time and limited to approximately 500 *user:ids* ($\max(U) \leq 500$).

2. Terminal: Equation (5) specifies the Franchise of a given terminal, F_t, as a set of *user:ids*. In ADEPT, F_t does not exist. One may define all the users for a given terminal, i.e., F_t; or alternatively, all the terminals for a given user. Because SYSLOG orders its tables by *user:id*, the latter definition was found more convenient to implement.
3. Job: The Franchise of a job is the *user:id* of the creator of the job at the time of LOGIN to the system. Currently, only one user has access to (and control of) a job ($\mu = 0$ for equation (6)).
4. File: Implementation of Franchise for a file (F_f), is more extensive than equation (7). In ADEPT, we wish to control not only who accesses a file, but also the quality of access granted. We have defined a set of four exclusive qualities of access, such that a given quality, q, is defined if

$$q \in \{\text{READ, WRITE, READ-AND-WRITE, READ-AND-WRITE-WITH-LOCKOUT-OVERRIDE}\} \quad (27)$$

ADEPT permits simultaneous access to a file by many jobs if the quality of access is for READ only. However, only one job may access a file with WRITE, or READ-AND-WRITE quality. ADEPT automatically locks out access to a file being written to avoid simultaneous reading and writing conflicts. A special access quality, however, does permit lockout override. Equation (7) can now be extended as a set of pairs,

$$F_f = \{(u_f^0, q^0), (u_f^1, q^1), \cdots, (u_f^\gamma, q^\gamma)\} \; ; \quad (28)$$

where q^i are not necessarily distinct and are given by equation (27).

The implementation of equation (28) is dependent upon γ, the number of franchised users. When $\gamma = 0$, we have the ADEPT Private file, exclusive to the owner, u_f^0; for $\gamma = \max(U)$, we have the Public file; values of γ between these extremes yield the Semi-Private file. γ is implicitly encoded as the ADEPT "privacy" item in the file's catalog control data, and takes the place of F_f for all cases except a Semi-Private file. For that case exclusively, equation (28) holds and an actual F_f list of *user:id*, *quality* pairs exists as a need-to-know list. The owner of a file specifies and controls the file's privacy, including the composition of the need-to-know list.

Security control initialization: SYSLOG

SYSLOG is a component of the ADEPT initialization package responsible for arming the security controls. It operates as one of a number of system start-up options prior to the time when terminals are enabled. SYSLOG sets up the security profile data for *user:id* and *terminal:id*, i.e., the "given constants" of Table I.

SYSLOG creates or updates a highly sensitive system disc file, where each record corresponds to an authorized user. These records are constructed from a deck of cards consisting of separate data sets for *compartment* definitions, *terminal:id* classification, and *user:id* clearance. The dictionary of *compartment* definitions contains the less-than-9-character mnemonic for each member of the Category set. Data sets are formed from the card types shown in Table II. Use of *passwords* is described later in the LOGIN procedure.

An IDT card must exist for each authorized user; the PWD, DEV, SEC, and CAT card types are optional. Other card types are possible, but not germane to security control, e.g., ACT for accounting purposes. More than one PWD, DEV, and CAT card is acceptable up to the current maximum data limits (i.e., 64 *passwords*, 48 *terminal:ids*, and 16 *compartments*).

A variety of legality checks for proper data syntax, quantity, and order are provided. SYSLOG assumes the following default conditions when the corresponding card type is omitted from each data set:

PWD	No *password* required
DEV	All *terminal:ids* authorized
SEC	A = UNCLASSIFIED
CAT	C = null (all zero mask)

This gives the lowest user clearance as the default, while permitting convenient user access. Various options exist in SYSLOG to permit maintenance of the internal SYSLOG tables, including the replacement or deletion of existing data sets in total or in part.

The sensitivity of the information in the security control deck is obvious. Procedures have been developed at each installation that give the function of deck creation, control, and loading to specially cleared security personnel. The internal SYSLOG file itself is protected in a special manner described later.

Access control

A fundamental security concern in multi-access systems is that many users with different clearances will be simultaneously using the system, thereby raising the

TABLE II—SYSLOG control cards

Card Type	*Purpose*
DICT	Identifies start of data set of *compartment* definitions.
compartment$_1$ ··· *compartment*$_{16}$	Defines up to 16 *compartments*.
TERMINAL	Identifies start of data sets of terminal definitions.
UNIT *terminal:id*	Identifies start of a terminal data set.
IDT *user:id*	Identifies start of a user data set.
PWD *password* ··· *password*	Defines legal *passwords* for *user:id* up to 64.
DEV *terminal:id*$_1$ ··· *terminal:id*$_{48}$	Defines legal terminals for *user:id* up to 48.
SEC *Authority*	Defines *user:id* Authority.
CAT *compartment*$_1$ ··· *compartment*$_{16}$	Defines *user:id* Category set.

possibility of security compromise. Since programs are the "active agents" of the user, the system must maintain the integrity of each and of itself from accidental and/or deliberate intrusion. A multifile system must permit concurrent access by one or more jobs to one or more on-line, independently classified files.

ADEPT is all these things—multiuser, multiprogram, and multifile system. Thus, this section deals with access control over users, programs, and files.

User access control: LOGIN

To gain admittance to the system, a user must first satisfy the ADEPT LOGIN decision procedure. This procedure attempts to authenticate the user in a fashion analogous to challenge-response practices.

The syntax of the ADEPT LOGIN command, typed by a user on his terminal, is as follows:

/LOGIN *user:id password accounting*

Figure 1 pictorially displays the LOGIN decision procedure based upon the user-specified input parameters. *User:id* is the index into the SYSLOG file used to retrieve the user security profile. If no such record exists (i.e., equation (22) fails), the LOGIN is unsuccessful and system access is denied. If the security profile is found, LOGIN next retrieves the *terminal:id* for the keyboard in use from internal system tables, and searches for a match in the *terminal:id* list for which the *user:id* was franchised by SYSLOG. An unsuccessful search is an unsuccessful LOGIN.

If the terminal is franchised, then the current *password* is retrieved from the SYSLOG file for this *user:id* and matched against the *password* entered as a keyboard parameter to LOGIN. An unsuccessful match is again an unsuccessful LOGIN. Furthermore, the terminal is ignored (will not honor input) for approximately 30 seconds to frustrate high-speed, computer-assisted, penetration attempts. If, however, the match is successful (equation (22) holds), the current *password* in the SYSLOG file for this *user:id* is discarded and LOGIN proceeds to create the job clearance.

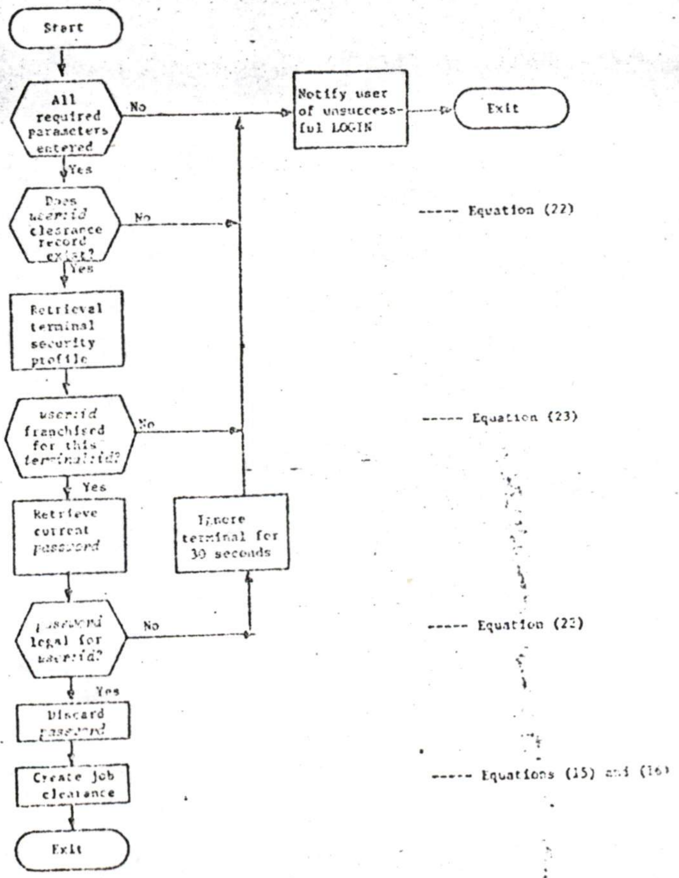

Start
All required parameters entered
No
Yes
Notify user of unsuccessful LOGIN
Exit
Does user:id clearance record exist?
No
Yes
----- Equation (22)
Retrieval terminal security profile
user:id franchised for this terminal:id?
No
Yes
----- Equation (23)
Retrieve current password
Ignore terminal for 30 seconds
password legal for user:id?
No
Yes
----- Equation (22)
Discard password
Create job clearance
----- Equations (15) and (16)
Exit

Figure 1—LOGIN decision procedure

Passwords in ADEPT obey the same syntax conventions as *user:id*. (See the earlier description of User Identification.) Although easily increased, currently SYSLOG permits up to 64 *passwords*. Each successful LOGIN throws away the user *password*; 64 successful LOGINs are possible before a new set of *passwords* need be established. If other than random, once-only *passwords* are desired, the 64 *passwords* may be encoded in some algorithmic manner, or replicated some number of times. Once-only *passwords* is an easily implemented technique for user authentication, which has been advocated by others.[2,7] It is a highly effective and secure technique because of the high permutability of 12-character-*passwords* and their time and order interdependence, known only to the user.

Once the authentication process is completely satisfied, LOGIN creates the job security profile according to equations (15) and (16) of our model. That is, the lower Authority of the user and the terminal becomes A_j, and the intersection (logical AND) of the user and terminal Category sets becomes the Category of the job, C_j. For example, a user with TOP SECRET Authority and a Category set (1001 1001 0000 1101) operating from a SECRET level terminal with a Category set (0000 0000 0000 0010) controls a job cleared to SECRET with an empty Category set.

Program access control: LOAD

As noted earlier, the ADEPT Executive consists of two parts: BASEX, the resident part, and EXEX, the swapped part. EXEX is a body of reentrant code shared by all users; however, it is treated as a distinct program in each user's job. Up to four programs can exist concurrently in the job. Each operates with the job clearance—the job clearance umbrella.

LOAD is the ADEPT component used to load the programs chosen by the user; it is part of EXEX and hence operates as part of the user's job with the job's clearance. Programs are cataloged files and as such may be classified with a given security profile. As is described in "File Access Control" below, LOAD can only load those programs for which the job clearance is sufficient. Once loaded, however, the new program operates with the job clearance.

In this manner, we see the power of the job umbrella in providing smooth, flexible user operation concurrent with necessary security control. Program files may be classified with a variety of security profiles and then operate with yet another, i.e., the job clearance. By this technique security is assured and programs of different classifications may be operated by a user as one job. It permits, for example, an unclassified program file (e.g., a file editor) to be loaded into a highly classified job to process sensitive classified data files.

File access control: OPEN

Before input/output can be performed on a file, a program must first acquire the file by an OPEN call to the Cataloger. Each program must OPEN a file for itself before it can manipulate the file, even if the file is already OPENed for another program. A successful OPEN requires proper specification of the file's descriptors—some of which are in the OPEN call, others of which are picked up directly by the Cataloger from the job environment area (e.g., job clearance, *user:id*)—and satisfactory job clearance and *user:id* need-to-know qualifications according to equations (25) and (26) of our model. Equation (25) is implemented as (8) as a straightforward magnitude comparison between A_j and A_f. Equation (25) is implemented as (9) as an equality test between C_f and $(C_j \wedge C_f)$. We use $(C_j \wedge C_f)$ to ensure that C_f is a subset of the job categories; i.e., the job umbrella. Lastly, equation (26) is a NOP if the file is Public; a simple equality test between u_j and u_f if the File is Private; and a table search of F_f for u_j if the file is Semi-Private. These tests do increase processing time for file access; however, the tests are performed only once at OPEN time, where the cost is insignificant relative to the I/O processing subsequently performed on the file.

The quality of access granted by a successful OPEN, and subsequently enforced for all I/O transfers, is that requested, even if the user has a greater Franchise. For example, during program debugging, the owner of a file may OPEN it for READ access only, even though READ-AND-WRITE access quality is permitted. He thereby protects his file from possible uncontrolled modification by an erroneous WRITE call.

Considerable controversy surrounds the issue of automatic classification of new files formed by subset or merger of existing files. The heart of the issue is the poor accuracy of many such classification techniques[17] and the fear of too many over-classified files (a fear of operations personnel) or of too many under-classified files (a fear of the security control officers). ADEPT finesses the problem with a clever heuristic—most new files are created from existing files, hence classify the new file as a private file with the composite Authority and Category of all files referenced. This is achieved in ADEPT by use of the "high-water mark."

Starting with the boundary conditions of equations (11) and (13), the Cataloger applies equations (12) and

(14) for each successful file OPEN, and hence maintains the composite classification history of all files referenced by the job. For each new and temporary file OPEN, the Cataloger applies equations (17), (18), and (19); they are reapplied for each CLOSE of a new file, to update the classification (due to changes in the high-water mark since the OPEN) when the file becomes an existing cataloged file in the inventory. The scheme rarely underclassifies, and tends to overclassify when the new file is created late in the job cycle, as shown by boundary equations (20) and (21).

Trans-formal security features

ADEPT contains a host of features that transcend the formalism presented earlier. They are described here because they are integral to the total security control system and form a body of experience from which new formalisms can draw.

Computer hardware

ADEPT operates on an IBM System 360/50 and is, therefore, limited to the hardware available. Studies by Bingham[9] suggest a variety of hardware features for security control, many of which are possessed by System 360.

IBM System 360 can operate in one of two states: the Supervisor state, or the Problem state. ADEPT executive programs operate in the Supervisor state; user programs operate in the Problem state.

A number of machine instructions are "privileged" to the Supervisor state only. An attempt to execute them in the Problem state is trapped by the hardware and control is returned to the executive program for remedial action. ADEPT disposes of these alarms by suspending the guilty job. (A suspended job may be resumed by the user.) Clearly, instructions that change the machine state are privileged to the executive only.

Another class of privileged instructions consists of those dealing with input/output. Problem state programs cannot directly access information files on secondary memory storage devices such as disc, tape, or drum. They must access these files indirectly by requests to the executive system. The requests are subjected to interpretive screening by the executive software.

Main memory is selectively protected against unauthorized change (write protected). We have also had the 360/50 modified to include fetch protection, which guards against unauthorized reading of—or executing from—protected memory. The memory protect instructions are also privileged only in the Supervisor state.

ADEPT software protects memory on a 4096-byte "page" basis (the hardware permits 2048-byte pages), allowing a non-contiguous mosaic of protected pages in memory for a given program. To satisfy multiprogramming, many different protection groups are needed. Through the use of programmable 4-bit hardware masks, up to 15 different protection groups can be accommodated in core concurrently. ADEPT executive programs operate with the all-zero "master key" mask, permitting universal access by all Basic and Extended Executive components.

There are five classes of interrupts processed by System/360 hardware: input/output, program, supervisor call, external, and machine check. Any interrupts that occur in the Problem state cause an automatic hardware switch to the Supervisor state, with CPU control flowing to the appropriate ADEPT executive interrupt controller. All security-vulnerable functions including hardware errors, external timer and keyboard actions, user program service requests, illegal instructions, memory protect violations, and input/output, are called to the attention of ADEPT by the System/360 interrupt system. The burden for security integrity is then one for ADEPT software.

Monitor software

Inducing the system to violate its own protection mechanisms is one of the most likely ways of breaking a multi-access system. Those system components that perform tasks in response to user or program requests are most susceptible to such seduction.

On-Line debugging

The debugging program provides an on-line capability for the professional programmer to dynamically look at and change selected portions of his program's memory. DEBUG can be directed to access sensitive core memory that would not be trapped by memory protection, since, as an EXEX component operating in the Supervisor state, DEBUG operates with the memory protection master key. To close this "trap door," DEBUG always performs interpretive checks on the legality of the debugging request. These checks are based upon address-out-of-bounds criteria, i.e., the requested debugging address must lie within the user's program area. If not, the request will be denied and the user warned, but he will not be terminated as has been suggested.[7]

Input/output

Input/output in System/360 is handled by a number of special-purpose processors, called Selector Channels. To initiate any I/O, it is necessary for a channel program to be executed by the Selector Channel.

SPAM, the BASEX component that permits symbolic input/output calls from user programs, is really a special-purpose compiler that produces I/O channel programs from the SPAM calls. These channel progams are subsequently delivered and executed by the ADEPT Input/Output Supervisor, IOS.

SPAM permits a variety of calls to read, write, alter, search for, and position to records within cataloged files. To achieve these ends, SPAM depends upon a variety of control tables dynamically created by the Cataloger in the job environment.

The initiating and subsequent monitoring of channel program execution is the responsibility of the BASEX Input/Output Supervisor, IOS. IOS is called to execute a channel program (EXCP). System components, such as SPAM, branch to IOS at a known entry point that is fetch-protected against entry in the Problem state. IOS is off-limits to user programs attempting to access cataloged storage. For protection against unauthorized EXCP requests, IOS always performs legality checks before executing a channel program. These checks begin by examination of the device addressed by the channel program. If it is the device address for cataloged storage, further checks are made to determine the machine state of the calling program. That state must be Supervisor state for the call to be honored. A call in the Problem state would indicate an illegal EXCP call from a user program.

IOS makes further checks to guarantee the validity of an I/O request [illegible] checks to see that the specified buffer areas for the [illegible] transfer do not overlay the channel program itself, and lie within the user's program memory area, i.e., do not modify or access system or protected memory.

Covert I/O violations are also forestalled since I/O components take direction from information stored in the job environment—an area read- and write-protected from Problem state programs.

Classified residue

Classified residue is classified information (either code or data) left behind in memory (i.e., core, drum, or disc) after the program that referenced it has been dismissed, swapped out, or quit from the system. The standard solution to the problem is to dynamically purge the contaminated memory (e.g., overwrite with random numbers, or zeros). In a system supporting over 1/4 billion bytes of memory, that solution is unreasonable and in conflict with high performance goals. ADEPT's solution to the dilemma of denying access to classified residue while maintaining high performance depends upon techniques of controlled memory allocation.

1. *Core Residue*

As noted earlier, all core storage is allocated as 4096-byte pages. These pages are always cleared to zero when allocated, thereby overwriting any potential residue.

Via the program's page map, the ADEPT executive system labels all code and data pages (they need not be contiguous) belonging to a given program with a single hardware memory protection key, thereby prohibiting unauthorized reading or writing by other, potentially co-resident user programs that may be in execution. Furthermore, BASEX keeps a running account of the status and disposition of all pages of core.

The Loader and Swapper components of ADEPT always work with full 4096-byte pages. Unfilled portions of pages at load time are kept cleared to zero as when they were allocated, and the full 4096 bytes are swapped into core, if not already resident, each scheduled time slice. Further, newly allocated pages are marked as "changed" pages, thus guaranteeing subsequent swap out to drum.

With these procedures, ADEPT denies access by a user or program to those pages of core not identified as part of his program, and clears core residue by over-writing accessible core at load and swap times.

2. *Drum Residue*

ADEPT always clears a drum page to zero before it is allocated. The page may subsequently be cleared again to user-specified data. ADEPT also maintains a drum map that notes the disposition of all drum pages (800 pages for the IBM 2303 drum). Drum input/output, like all ADEPT I/O, is controlled by executive privileged instructions.

3. *Disc Residue*

Disc files in ADEPT are maintained as "dirty" memory. That is, the large capacity of the file system makes it infeasible to consider automatic over-writing techniques for residue control; therefore, deleted disc tracks are returned to the available storage pool contaminated and unclean. It then becomes the burden of the

ADEPT file system to control any unauthorized file access, whether to cataloged files or uncataloged disc memory.

Team work between the Cataloger, SPAM and IOS components of ADEPT achieves this control via legality checking of all OPEN and I/O requests.

For example, all disc packs are labeled internally and externally with their *volume:id*, and this label is checked at the time of mounting by the Cataloger OPEN procedure to assure proper volume mounting. Tapes may also be labeled and checked as a user option.

Of particular note, SPAM always assumes that an end-of-file (EOF) immediately follows the last record written in a new file, and it prohibits reading beyond that EOF. Contaminated tracks allocated to new files cannot be read until they are first written. The act of writing advances the EOF and the user simultaneously over-writes the classified residue with his own data. The user cannot skip over the EOF, and the EOF location is itself protected in the job environment area.

4. *Tape Residue*

No special features for tape residue control are implemented in ADEPT. Tape residue control is easily satisfied by manual, off-line tape degaussing prior to ADEPT use.

System files

Equation (28) led us to examine Private, Semi-Private, and Public files. ADEPT possesses two additional file privacies that transcend our model; both are system files. Privacy-4 system files are the need-to-know lists created by the Cataloger itself for Semi-Private files. Privacy-5 system files are private system memory for the SYSLOG files and the catalogs themselves.

Access to these files is restricted to the system only. Special access checks are made that differ from those of equations (25) and (26). First, a special *user:id* is required that is not a member of U (i.e., not in the SYSLOG file). Second, the program making the OPEN call must be in Supervisor state. Third, the program making the OPEN call must be a member of a short list of EXEX programs. The list is built into the Cataloger at the time of compilation. In this manner, access to system files is severely restricted, even to system programs.

Security service commands

ADEPT provides a variety of service commands that involve security control. The commands are listed in Table III. Note that commands VARYON, VARYOFF, REPLACE, LISTU, AUDIT, AUDOFF, and WRAP-UP are restricted to a particular terminal—the Security Officer's Station.

TABLE III—Security service commands

Command	*Purpose*
AUDIT*	Turns on security audit recording.
AUDOFF*	Turns off security audit recording.
CHANGE	Enables the owner of a file to change any of the access control information of the file.
CREATE	Enables a user to create a Semi-Private file and its need-to-know list.
LISTU*	Lists by *terminal:id* all the current logged in *user:ids*.
RECLASS	Enables a user to raise or lower his job clearance between the bounds of the original LOGIN and current high-water mark clearance.
RELOG	Like LOGIN, but reconnects a user to an already existing job, as when a remote terminal drops off the communications line.
REPLACE*	Enables a user to move his job to another terminal or to reclassify a given device.
SECURITY	Print on the user's terminal approximately every 100 lines (or only by request) the job high-water mark (or clearance by request) as a reminder to the user and as a classification stamp of the level of current security activity.
VARYON/VARYOFF*	Permits terminals to be varied on- and off-line for flexibility in system maintenance and configuration control.
WRAPUP*	Shuts down system after a specified elapsed time.

* Restricted to Security Officer's Station only.

Audit

The AUDIT function records certain transactions relating to files, terminals, and users, and is the electronic equivalent of manual security accountability logs. Its purpose is to provide a record of user access in order to determine whether security violations have occurred and the extent to which secure data has been compromised. The AUDIT function may be initiated only at start-up time, but may be terminated at any time. All data re recorded on disc or tape in real time so the data is safe if the system malfunctions. An auxiliary utility program, AUDLIST, may be used to list the AUDIT file. The information recorded is shown in Table IV.

Implementation of AUDIT is quite straightforward, a product of general ADEPT recording and instrumentation.[18,19] AUDIT is an EXEX component that is called by, and at the completion of, each function o be recorded. The information to be recorded is pass d to AUDIT in the general registers. Additional I/O overhead is the primary cost incurred in the operation of AUDIT, for swapping and file maintenance. This cost is nominal, however, amounting to less than one percent of the CPU time.

SUMMARY

In summary we may ask: How well have we met our goals? First, we believe we have developed and success-

TABLE IV—Security events and information audited by ADEPT-50

EVENT	TIME	STATUS	JOB SECURITY PROFILE	USER SECURITY PROFILE	ACCOUNT NUMBER	USER:ID	TERMINAL:ID	CPU TIME	NEW TERMINAL	TERMINAL SECURITY PROFILE	FILE NAME	FILE OWNER ID	FILE SECURITY PROFILE	FILE FORM	FILE VOLUME NUMBER	PROSE CATEGORY NAMES
LOGIN	X	X	X	X	X	X	X									
LOGOUT	X	X						X								
OPEN FILE	X	X									X	X	X	X	X	
REOPEN[1] FILE	X	X									X	X	X	X	X	
CHANGE FILE	X	X									X	X	X	X	X	
CLOSE FILE	X	X									X	X		X		
DELETE FILE	X	X									X	X		X		
RECLASS	X	X	X													
REPLACE	X	X					X		X	X						
DEVICE LIST[2]	X									X						
CATEGORY DICTIONARY[3]	X															X
RESTART[4]	X															
WRAPUP[5]	X															

1 This is the "OPEN existing file" command.

2 A list of all the terminal devices and their assigned security and categories is recorded at each system load.

3 A list of the prose category names is recorded at each system load.

4 Whenever the system is restarted on the same day (and AUDIT had been turned on earlier that day) the time of the restart is recorded.

5 The time that the AUDOFF action was taken, or the time that the WRAPUP function called AUDIT, to terminate the AUDIT function.

fully demonstrated a security control mechanism that more than adequately supports heterogeneous levels and types of classification. Of note in this regard is the LOGIN decision procedure, access control tests, job umbrella, high-water mark, and audit trails recording. The approach can be improved in the direction of more compartments (on the order of 1000 or more), extension of the model to include system files, and the implementation of a single Franchise test for all security objects. The implementation needs redundant encoding and error detection of security profile data to increase confidence in the system—though we have not ourselves experienced difficulty here. The increase in memory requirements to achieve these improvements may force numerical encoding of security data, particularly Category, as suggested by Peters.[7]

Second, SYSLOG has been highly successful in demonstrating the concept of "security arming" of the system at start-up time. Our greatest difficulty in this area has been with the human element—the computer operators—in preparing and handling the control deck. In opposition to Peters,[7] we believe the operator should not be "designed out of the operation as much as possible," but rather his capabilities should be upgraded to meet the greater levels of sophistication and responsibility required to operate a time-sharing system.[10] He should be considered part of line management. ADEPT is oriented in this direction and work now in progress is aimed at building a real-time security surveillance and operations station (SOS).

Third, we missed the target in our attempt to isolate and limit the amount of critical coding. Though much of the control mechanism is restricted to a few components—LOGIN, SYSLOG, CATALOGER, AUDIT—enough is sprinkled around in other areas to make it impossible to restrict the omnipotent capabilities of the monitor, e.g., to run EXEX in Problem state. Some additional design forethought could have avoided some of this dispersal, particularly the wide distribution in memory of system data and programs that set and use these data. The effect of this shortcoming is the need for considerably greater checkout time, and the lowered confidence in the system's integrity.

Lastly, on the brighter side, we were surprisingly frugal in the cost of implementing this security control mechanism. It took approximately five percent of our effort to design, code, and checkout the ADEPT security control features. The code represents about ten percent of the 50,000 instructions in the system. Though the code is widely distributed, SYSLOG, security commands, LOGIN, AUDIT, and the CATALOGER account for about 80 percent of it. The overhead cost of operating these controls is difficult to measure, but it is quite low, in the order of one or two percent of total CPU time for normal operation, excluding SYSLOG. (SYSLOG, of course, runs at card reader speed.) The most significant area of overhead is in the checking of I/O channel programs, where some 5 to 10 msec are expended per call (on the average). Since this time is overlapped with other I/O, only CPU bound programs suffer degredation. AUDIT recording also contributes to service call overhead. In actuality, the net operating cost of our security controls may be zero or possibly negative, since AUDIT recordings showed us numerous trivial ways to measurably lower system overhead.

ACKNOWLEDGMENTS

I would like to acknowledge the considerable encouragement I received in the formative stages of the ADEPT security control design from Mr. Richard Cleaveland, of the Defense Communications Agency (DCA). I would like to thank Mrs. Martha Bleier, Mr. Peter Baker, and Mr. Arnold Karush for their patient care in designing and implementing much of the work I've described. Also, I wish to thank Mr. Marvin Schaefer for assisting me in set theory notation. Finally, I would like to applaud the ADEPT system project personnel for designing and building a time-sharing system so amenable to the ideas discussed herein.

REFERENCES

1 A HARRISON
The problem of privacy in the computer age: An annotated bibliography
RAND Corp Dec 1967 RM-5495-PR/RC

2 L J HOFFMAN
Computers and privacy: A survey
Stanford Linear Accelerator Center Stanford Univ Aug 1968 SLAC-PUB-479

3 H E PETERSEN R TURN
System implications of information privacy
Proc SJCC Vol 30 1967 291-300

4 W H WARE
Security and privacy in computer systems
Proc SJCC Vol 30 1967 279-282

5 W H WARE
Security and privacy: Similarities and differences
Proc SJCC Vol 30 1967 287-290

6 R LINDE C WEISSMAN C FOX
The ADEPT-50 time-sharing system
Proc FJCC Vol 35 1969 Also issued as SDC Doc SP-3344

7 B PETERS
Security considerations in a multi-programmed computer system
Proc SJCC Vol 30 1967 283-286

8 RYE CAFRI COINS OCTOPUS SADIE Systems

NOC Workshop National Security Agency Oct 1968
9 H W BINGHAM
Security techniques for EDP of multi-level classified information
Rome Air Development Center Dec 1965 RADC-TR-65-415
10 R M GRAHAM
Protection in an information processing utility
ACM Symposium on Operating Systems Principles Oct 1967 Gatlinburg Tenn
11 L J HOFFMAN
Formularies—Program controlled privacy in large data bases
Stanford Univ Working Paper Feb 1969
12 D K HSIAO
A file system for a problem solving facility
Dissertation in Electrical Engineering Univ of Pa 1968
13 J I SCHWARTZ C WEISSMAN
The SDC time-sharing system revisited
Proc ACM Conf 1967 263-271
14 P BARAN
On distributed communications: IX, security, secrecy, and tamper-free considerations
RAND Corp Aug 1964 RM-3765-PR
15 C WEISSMAN
Programming protection: What do you want to pay?
SDC Mag Vol 10 No 8 Aug 1967
16 J P TITUS
Washington commentary—Security and privacy
CACM Vol 10 No 6 June 1967 379-380
17 I ENGER et al
Automatic security classification study
Rome Air Development Center Oct 1967 RADC-TR-67-472
18 A KARUSH
The computer system recording utility: Application and theory
System Development Corp March 1969 SP-3303
19 A KARUSH
Benchmark analysis of time-sharing systems: Methodology and results
System Development Corp April 1969 SP-3343
20 R R LINDE P E CHANEY
Operational management of time-sharing systems
Proc 21st Nat ACM Conf 1966 149-159

The ADEPT-50 time-sharing system

by R. R. LINDE and C. WEISSMAN

System Development Corporation
Santa Monica, California

and

C. E. FOX

King Resources Company
Los Angeles, California

INTRODUCTION

In the past decade, many computer systems intended for operational use by large military and governmental organizations have been "custom made" to meet the needs of the particular operational situation for which they were intended. In recent years, however, there has been a growing realization that this design approach is not the best method for long term system development. Rather, the development of general purpose systems has been promoted that provide a broad, general base on which to configure new systems. The concepts of time-sharing and general-purpose data management have been under development for several years, particularly in university or research settings.[1,2,3] These methods of computer usage have been tested, evaluated, and refined to the point where today they are ready to be exploited by a broad user community.

Work on the Advanced Development Prototype (ADP) contract was begun in January 1967 for the purpose of demonstrating—in an operational environment—the potential of automatic information-handling made possible by recent advances in computer technology, particularly advances in time-sharing executives and general-purpose data management techniques. The result of this work is a large-scale, multi-purpose system known as ADEPT, which operates on IBM system 360 computers.*

The entire ADEPT system is now being used at four field installations in the Washington, D. C. area, as well as at SDC in Santa Monica. The system was installed at the National Military Command System Support Center in May 1968, at the Air Force Command Post in August 1968, and at two other government agencies in January 1969. These four field sites collectively run ADEPT from 80 to 100 hours per week, providing a total of some 2000 terminal hours of time-sharing service monthly to their users.

The ADEPT system consists of three major components: a time-sharing executive; a data management system adapted from SDC's Time-Shared Data Management System (TDMS) described by Bleier,[4] and a programmer's package. This paper deals exclusively with the ADEPT Time-Sharing Executive, and particularly with the more novel aspects of its architecture and construction. Before examining these aspects it will be instructive if we review the basic design and hardware configuration of the system.

A general purpose operating system

The ADEPT executive is a general-purpose time-

* Development of ADEPT was supported in part by the Advanced Research Projects Agency of the Department of Defense.

sharing system. The system operates on a 360 Model 50 with approximately 260,000 bytes of core memory, 4 million bytes of drum memory, and over 250 million bytes of disc memory, shown graphically in Figure 1 and schematically in the appendix. With this machine configuration, ADEPT is designed to provide responsive on-line interactive service, as well as background service to approximately 10 concurrent, user jobs. It handles a wide variety of different, independent application programs, and supports the use of large random-access data files. The design—basically a swapping system—provides for flexibility and expansion of system functions, and growth to more powerful models in the 360 family.

ADEPT functions both as a batch processor (whereby jobs are accumulated and fed to the CPU for operation one by one) and as an interactive, on-line system (in which the user controls his job directly in real time simply by typing console requests).

Viewed as a batch system, ADEPT allows jobs to be submitted to console operators or submitted from consoles via remote batch commands (remote job entry). In either case, jobs are "stacked" for execution by ADEPT in a first-in/first-out order. The stack is serviced by ADEPT as a background task, subject to the priorities of the installation and the demands of "foreground" interactive users. Viewed as an interactive system, ADEPT allows the user to work with a typewriter, allowing computer-user dialog in real time. Via ADEPT console commands, the user identifies himself, his programs, and his data files, and selectively controls the sequence and extent of operation of his job in an ad lib manner. A prime advantage of the interactive use of ADEPT is that the system provides an extendable library of service programs that permit the user to edit data files, compile or assemble programs, debug and eliminate program errors, and generally manage large data bases in a responsive on-line manner.

System architecture

The architecture of the ADEPT executive is that of the "kernel and the shell". The "kernel," referred to as the Basic Executive (BASEX), handles the major problems of allocating and scheduling hardware resources. It is small enough to be permanently resident in low core memory, permitting rapid response to urgent tasks, e.g., interrupt control, memory allocation, and input/output traffic. The "shell," referred to as the Extended Executive (EXEX), provides the interface between the user's application program and the "kernel". It contains those non-urgent, large-

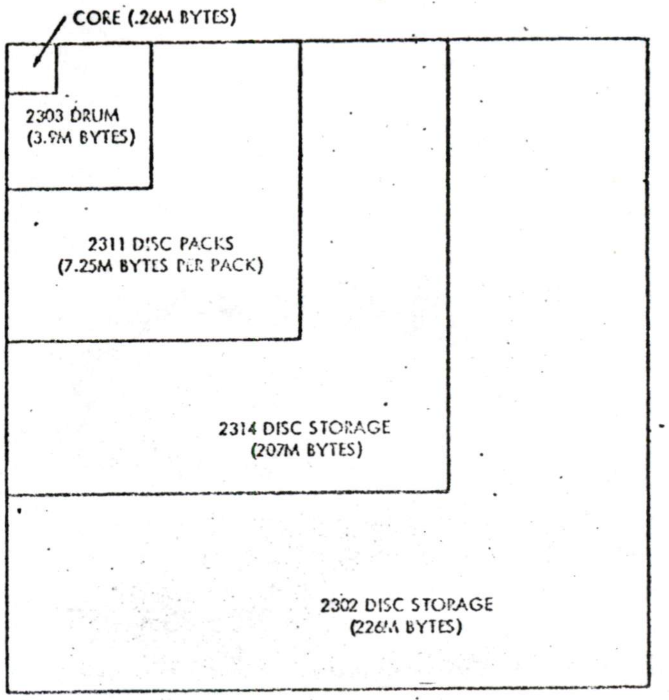

CORE (.26M BYTES)
2303 DRUM (3.9M BYTES)
2311 DISC PACKS (7.25M BYTES PER PACK)
2314 DISC STORAGE (207M BYTES)
2302 DISC STORAGE (226M BYTES)

Figure 1—Relative capacity of various ADEPT direct-access storage media available in less than 0.2 seconds. The initial system that operates at SDC utilizes core, 2303 drum, 2311 and 2314 disc packs, and 2302 disc storage. The NMCSSC system utilizes 2314 disc storage in lieu of 2311 or 2302 discs. The architecture of the ADEPT executive is such that it permits any combination of the above types of disc storage in varying amounts

task extensions of the basic "kernel" processes that are user-oriented rather than hardware-oriented; they may, therefore, be scheduled and swapped.

The version of the ADEPT time-sharing system thus far developed has multiple levels of control beyond the two-level "kernel-shell" structure—i.e., it can be thought of figuratively as an "onion skin". Figure 2 shows these relationships graphically.

Beyond EXEX, "object systems" may exist as subsystems of ADEPT (developed by the user community without modification to EXEX or BASEX), thus further distributing and controlling the system resources for the object programs that form still another level of the system. The design ideas embodied in ADEPT parallel those of Dijkstra,[5] Corbato,[6] and Lampson,[7] but differ in techniques of implementation.

The ADEPT Basic Executive operates in the lower quarter of memory, thereby providing three quarters of memory for user programs. With the current H core configuration, ADEPT preempts the first 65,000 bytes of core memory, the bulk of which is dedicated to BASEX; EXEX must then operate in user memory

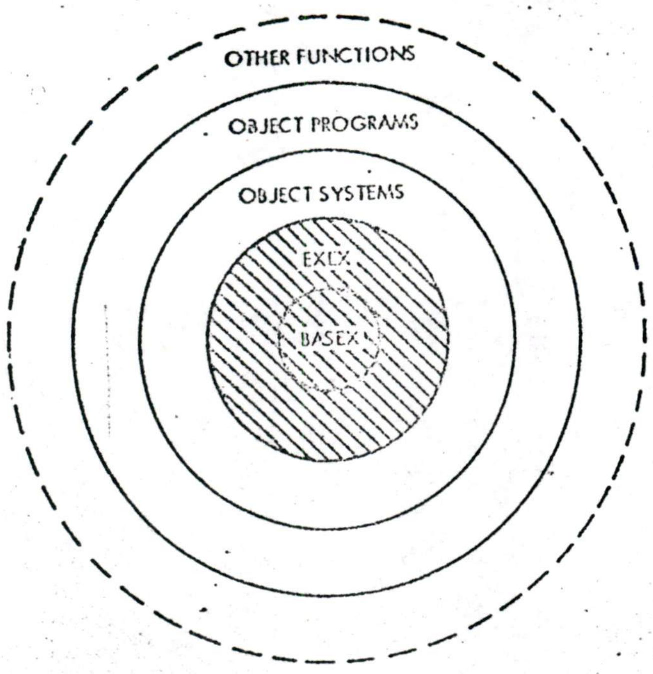


Figure 2—Multiple levels of control in ADEPT

in a fashion similar to user programs. ADEPT is designed to operate itself and user programs as a collection of 4096-byte pages. BASEX is identified as certain pages that are fixed in main storage and that cannot be overlayed or swapped. EXEX and other programs are identified as sets of pages that move dynamically between main storage and swap storage (i.e., drum). It is necessary to maintain considerably more descriptive information about these swappable programs than about BASEX. This descriptive information is carried in a set of system tables that, at any point in time, describe the current state of the system and each program.

ADEPT views the user as a job consisting of some number of programs (up to four for the 360/50H configuration) that were loaded at the user's request. These programs may be independent of one another or, with proper design, different segments of a larger task. Implicitly, EXEX is considered to be one of these programs. To simplify system scheduling, communication, and control, only one program in the user's set may be active (eligible to run) at a time. When ADEPT scheduling determines that a job may be serviced, the current job in core is saved on swap storage, and the active program of the next job is brought into core from swap storage and executed for a maximum period of time, called a quantum. The process then repeats for other jobs. Figures 3 and 4 schematically depict these relationships.

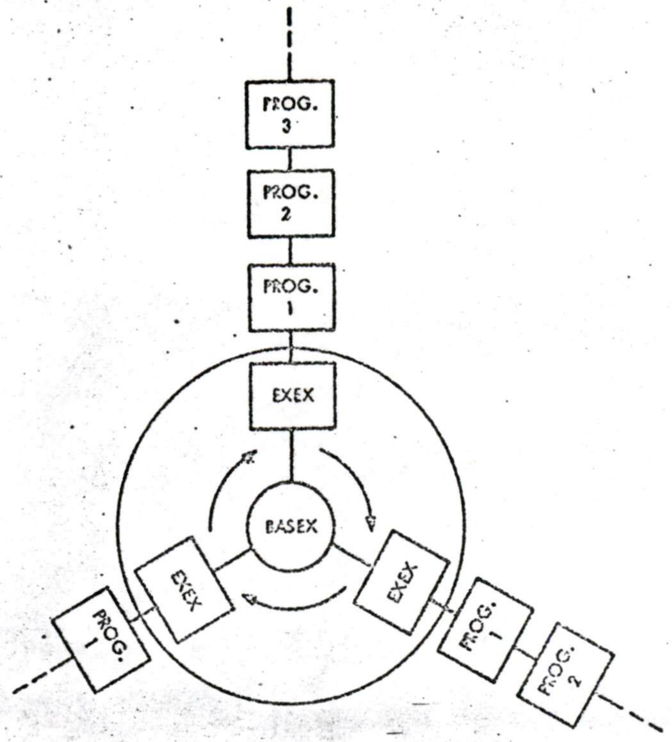


Figure 3—Simple commutation of users programs. This figure illustrates the relationship between user's programs' EXEX and BASEX. Each spoke represents a user's job, with his EXEX providing the interface between BASEX and the hardware resources. The maximum number of interactive job the IBM 360/50H configuration is ten.

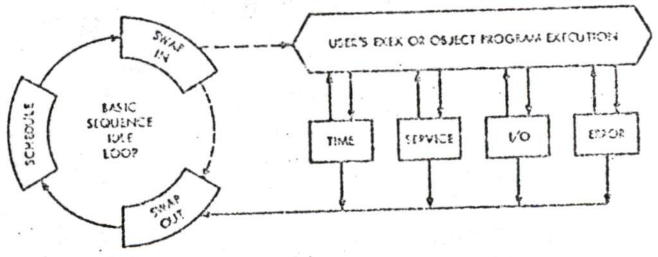


Figure 4—ADEPT's basic sequence of operation. This figure shows the basic operating system cycle: idle loop is interrupted by an external interrupt (an activity request); a program is scheduled, swapped into core from the drum, and executed escape from the execution phase occurs when quantum termination condition (e.g., time expiration, service or I/O call, error condition) is met; the program is then swapped out and control is returned to the idle loop (if no other programs are eligible to be scheduled).

Basic executive (BASEX)

Table I lists the BASEX components and their general functions as of the eighth and latest executive release. These basic system components form an integrated, non-reentrant, non-relocatable, permanently-resident, core memory package 16 pages long (each page is 4096 bytes). They are invoked by hardware interrupts in response to service requests by users of terminals and their programs. Note the division of input/output control into cataloged (SPAM and IOS), terminal (TWRI), and drum (BXEC) activities to permit local optimization for improved system performance.

TABLE I—Basic executive components

Component	*Function*
ALLOC	Drum and core memory allocation.
BXBUG	Debugger for executive programs.
BXEC	Basic sequence and swap control.
BXECSVC	SVC handlers for WAIT, TIME, DEVICE, STOP AND DISMISS calls.
EXEX	Linkage routines for EXEX (BASEX/EXEX interfaces); also services commands DIALOFF, DIALON.
INTRUP	First-level interrupt control.
IOS	Channel-program level input/output supervisory control.
RECORD	Records SVC, interrupt activity in BASEX.
SKED	Scheduler.
SPAM	Input/output access methods to cataloged storage.
TWRI	Terminal input/output control.
System Tables	Resident system data areas for communication table (COMTAB), logged-in user's table (JOB), loaded programs table (PQU), drum and core status tables (DSTAT, CSTAT), and a variety of other tables.

Extended executive (EXEX)

Unlike the tight, closed package of integrated BASEX components, EXEX is a loose, open-ended collection of semiautonomous programs. Table II lists this collection of programs. EXEX is treated by BASEX as a user program, with certain privileges, and each user is given his own "copy" of the EXEX. It is transparent to the user that EXEX is reentrant

TABLE II—Extended executive components

Component	*Function*
AUDIT	Maintains a real-time recording of all security transactions as an accountability log.
BMON	Batch monitor for control of background job execution.
CAT	Cataloger for file storage access control; also services FORGET command.
DTD	Transfers recording information from drum to disc.
DBUG	Debugger for non-executive (user) programs.
LOGIN	User authentication and job creation.
SERVIS	Library of service commands that are reentrant, interruptible and scheduled: APPEND, CHANGE, CREATE, CYLS, DELETE, DRIVES, INIT, LISTF, LISTU, LOAD, LOADD, LOAD and GO, OVERLAY, REPLACE, RESTORE, RESTORED, SAVE, SEARCH, VARYOFF, VARYON.
RUN	Remote batch job submission control servicing commands RUN and CANCEL.
XXTOO	Library of small, fast, executive service commands: CPU, BGO, BQUIT, BSTOP, DIAL, DRUMS, GO, LOGOUT, QUIT, RESTART, SKED, SKEDOFF, STATUS, STOP, TIME, USERS.
SYSDEF	Defines input/output hardware configuration at time of system start up.
SYSLOG	Defines authorized user/terminal security profiles at time of system start up.
TEST	Initializes system tables at time of system start up.
SYSDATA	Non-resident, shared, system data table for dial messages and other common data, e.g., lists of all logged-in users; other non-resident, job-specific tables also exist, e.g., job environment page, push-down list data page.

and is being shared with other users, except for its data space. Each job has its own "machine state" tables saved in its unique set of environment pages. This structure permits flexible modification and orderly system expansion in a modular fashion. EXEX is always scheduled in the same way as other user programs.

Though EXEX components are, in large part, non-self-modifying reentrant routines and thus, could at small cost, be relocatable; neither user programs nor EXEX components are relocated between swaps. The lack of any mapping hardware on the IBM 360/50 and the design goal and knowledge that most user programs would be of maximum size made unnecessary a software provision to relocate programs dynamically. User programs may be relocated once at load time, however.

Communication and control techniques used in ADEPT

Communication is the generic term used to cover those services that permit two (or more) programs to intercommunicate, be they system program, user program, or both. From this communication vantage point we shall examine the connective mechanism used between the Basic and Extended Executives; the techniques that allow components within the EXEX to make use of one another; and the system design that permits an object program to control its own behavior as well as to communicate with the system and with other object programs.

The ADEPT job or process

Before we discuss the system mechanics, let us examine how the system treats each user logically. A user in the system is assigned a job number. Each job in the system may be viewed as a separate *process*, and each process is, by definition, independent of all other processes running on the machine. A process—or job—is not a program. It is the logical entity for the execution of a program on the physical processor, and it may contain as many as four separate programs. A program consists of the set of machine instructions swapped into the processor for execution, and the Extended Executive is one of these programs.

The ADEPT executive requires a large number of system tables to permit Basic and Extended Executive communication. Conceptually, the use of descriptive tables defining the condition of a user's process is analogous to the state vector (or state word) discussed by Lampson and Saltzer.[8,9] That is, the collection of information contained by these tables is sufficient to define an inactive user's process state at any given moment. By resetting the central processor from the state vector, a user's job proceeds from an inactive to an active state as if no interruption had occurred. The state vector contains such items as the program counter, the processor's general registers, the core and drum map of all the programs in the job, and the peripheral storage file data. All of the collective data for each program or task in the process are contained in the state vector.

Basic and extended executive communication

Each ADEPT user (i.e., any person who initiates some activity within the system by typing in commands) is given a job number and assigned an entry in the JOB table. The JOB table contains the system's top-level bookkeeping on user activity. It contains the user's identification, his location, his security clearance, and a pointer to his program queue. Each user is assigned one entry, or JOB, in the table. Associated with each JOB are the one or more programs that the user is running.

Top-level bookkeeping on programs is contained in the Program Queue (PQU) table. Each PQU entry contains a program identification and some (but not all) information that describes that program in terms of its space requirements, its current activity, its scheduling conditions, and its relationship to other programs in the PQU that belong to the same JOB. The detailed descriptive information and the status of each JOB and its programs are carried in the swappable environment space.

The environment pages (there can be as many as four) comprise a number of separate tables that contain such information as the contents of the general registers, the swap storage page numbers where the balance of the program resides, the program map, and lists of all active data files. A single environment page (or pages) is shared by all programs that belong to the same JOB (user). The system design allows for environment page overflow at which time additional pages are assigned dynamically. The environment pages, PQU table, JOB table, and data pages comprise the state vector of the user's job.

To permit storage of "global" system variables, and to allow system components to reference system data that may be periodically relocated, there exists a system communication table, which resides in low core so that it can be referenced without loading a base register.

The IBM 360 supervisor call (SVC) is used exclusively by EXEX components and object programs to request BASEX services. Though additional overhead is incurred in the handling of the attendant interrupt, the centralization of context switching provided is of considerable value in system design, fabrication, and checkout.

Extended executive communication

An EXEX may make use of another EXEX function by use of the SVC call mechanism. To support the recursive EXEX, an additional SVC processing routine is required to manage the different recursive contexts. This routine, called the SVC Dispatcher, processes calls from user and EXEX functions alike, manages a swappable data page, and switches to an interface linkage routine. The data page contains a system communication stack that consists of a program's general registers and the Program Status Word at the time of the SVC. This technique is analogous to the push-down logic of recursive procedure calls found in ALGOL or LISP language systems. The stack provides a convenient means of passing parameters between routines in the EXEX. Since each job has its own unique data page and environment page, EXEX is both recursive and reentrant.

The environment status table (ESTAT) contains the swap and core location for each component in the EXEX and for eac' program in the job. It resides in the job environment page. When an EXEX service is requested, only that particular EXEX program is brought in from swap storage, rather than the full service library. The interface linkage routine provides this management function; it lies as a link between the SVC Dispatcher and the particular EXEX function. The interface routine picks up necessary work pages for the EXEX component involved and branches to that component after it is brought into core. The interface routine maintains a separate push-down stack of return addresses providing the means for the EXEX component to properly exit and return control to its interface routine and then to the system.

The EXEX component called may make additional EXEX SVC calls before exiting. To provide correct work page allocation during recursive calls, the interface routine also saves the work page core and drum page addresses in the push-down stack. Upon completion of a call, the EXEX component returns to its interface routine; the interface routine releases all allocated work pages to the system and branches to a common unwind procedure.

The unwind procedure, like the SVC Dispatcher, is simply a switching mechanism. It determines, via the stack, whether to return to a still higher level EXEX function, or to turn the EXEX off and exit to the Basic Sequence. This recursive/reentrant control is the most complex portion of ADEPT and is the "glue" that binds BASEX and EXEX together. Figure 5 illustrates the recursive process.

Object program communication

One of the more stringent services required of an operating system is the rapid interchange of large quantities of data between object programs. The interchange of even simple arrays, matrices, and tables via stack parameters or a common file suffers from the inadequacy of limited capacity or extensive I/O time. Many operating systems ignore this requirement, thereby restricting the general-purpose applications. Yet there are solutions to this problem, and one successful technique employed in the ADEPT system is that of "shared memory". Shared memory is achieved by using the basic mechanism for managing reentrancy, namely the program environment page map. Through the ADEPT SHARE Page call, an object program can request that designated pages of another program

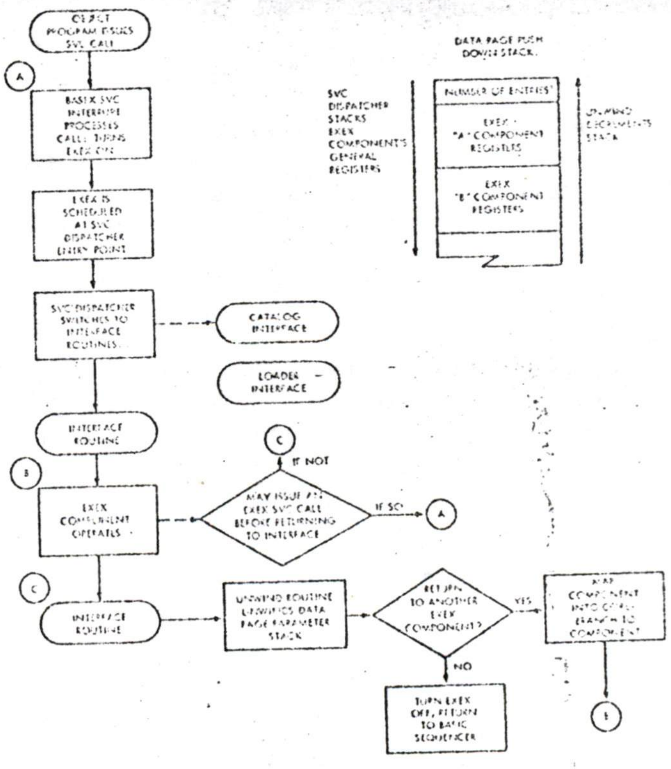

OBJECT PROGRAM ISSUES SVC CALL
A
BASEX SVC INTERRUPT PROCESSES CALL; TURNS EXEX ON
EXEX IS SCHEDULED AT SVC DISPATCHER ENTRY POINT
SVC DISPATCHER SWITCHES TO INTERFACE ROUTINES
CATALOG INTERFACE
LOADER INTERFACE
INTERFACE ROUTINE
B
EXEX COMPONENT OPERATES
C
IF NOT
MAY ISSUE AN EXEX SVC CALL BEFORE RETURNING TO INTERFACE
IF SO
A
C
INTERFACE ROUTINE
UNWIND ROUTINE UNWINDS DATA PAGE PARAMETER STACK
RETURN TO ANOTHER EXEX COMPONENT?
YES
MAP COMPONENT INTO CORE; BRANCH TO COMPONENT
NO
TURN EXEX OFF, RETURN TO BASIC SEQUENCER
B
DATA PAGE PUSH DOWN STACK
SVC DISPATCHER STACKS EXEX COMPONENT'S GENERAL REGISTERS
NUMBER OF ENTRIES
EXEX "A" COMPONENT REGISTERS
EXEX "B" COMPONENT REGISTERS
UNWIND DECREMENTS STACK

Figure 5—Block diagram of EXEX behavior and control

in the job be added to its map. If core page numbers are passed as parameters in various service calls, whole pages of data may be passed between programs. EXEX and many object programs operating under this system use this method for inter-program communication.

ADEPT operating on the IBM 360/50H restricts its user programs to 46 active core pages. However, by utilizing the GETPAGE call, an object program may acquire up to 128 drum pages and may subsequently activate and deactivate various page sets by utilizing another service call, ACTDEACT (activate/ deactivate). This scheme permits bulk data from disc storage to be placed on drum and operated upon at "swap" speeds. Thus skilled system users can achieve efficient use of time and memory by managing their own "paging". We consider this the best alternative considering the questionable state of other, automatic paging algorithms.[10,11,12,13] Most EXEX components use these calls for just such purposes. For example, the interface routines mentioned above use activate calls to "turn on" called components of the EXEX.

The Allocator component of ADEPT manages the page map for each program. This software map reflects the correspondence between drum and core pages, established initially by the SERVIS (service) component at load time. The Allocator's function is to inventory available core and drum pages by maintaining two resident system tables: one for core, the other for drum. Whe ever drum pages are released or obtained, the Allocator updates the page map in the job's environment page. The Allocator processes the SHARE (page), GETPAGE, FREEPAGE, and ACTDEACT calls from EXEX and object programs. SERVIS allows a program at run time to add data pages or to overlay program segments from disc or tape. In so doing, SERVIS makes use of the various Allocator calls.

Simulating console commands

An important attribute of ADEPT time-sharing is that nearly all the functions and services that can be initiated at the user's console can also be called forth within a user's program. A program designer can, for example, build a system of programs, which can operate in batch mode under the control of a program by issuing internal commands in much the same manner as the user sitting at the console. With this approach, the ADEPT batch monitor controls background tasks by simulating user terminal requests. Batch requests can be enqueued by users from any console and then processed in turn by this supervisor function.

Armed interrupts and rescue function

The basic design of ADEPT conveniently provides for processing object program "armed" interrupt calls. This means that an object program is able to conditionally start (wakeup) and stop (sleep) the execution of its own programs, and others as well. The conditions for employing wakeup calls include too much elapsed time, or the occurrence of unpredictable but anticipated events, e.g., errors and other program calls. In "arming" these "software-interrupt" conditions by object program calls, the program entry point(s) for the various conditions are specified. When such conditions occur, the operating system transfers to the specified entry point and gives the appropriate condition code. (Note that if we take this call one step further, and permit one object program to arm the software and hardware interrupts of another object program, we have the basic control mechanism necessary to permit the operation of "object systems.
necessary to permit the operation of "object systems," i.e., subexecutives—another level in the "onion skin" of ADEPT control.)

User programs interface with the ADEPT system primarily via the supervisor call (SVC) instruction; a secondary interface is provided via the program check interrupt that protects the program and system after various error conditions. The executive design allows user programs to trap all such interfaces with the system via its rescue arming mechanism. This means that one program can trap and get first-level control of all occurrences of SVC's and program checks within a single job. This mechanism also means, then, that the responsibility and meaning for these interfaces can be redefined at the user program level.

As of this writing, this mechanism is being employed to construct object systems for an improved batch monitor, an interface for the proposed ARPA Network,[14] and to experiment with automatic translators for compatibility with other operating systems. Other uses include improvements in program recovery in a variety of user tools, e.g., compiler diagnostics.

Resource allocation, access, and management

ADEPT system design, of course, includes a complete set of resource controls that monitor secondary storage devices.

The cataloger

The Cataloger, an EXEX component, is functionally analogous to the core/drum Allocator, but is used for devices accessible by user programs. It maintains an inventory of all assignable storage devices, assigns unused storage on the devices, maintains descriptions of the files placed on these devices, controls access to these files, and—upon authorized request—deletes any file. Specifically, the Cataloger:

- Assigns storage on 2302, 2311 and 2314 discs.
- Assigns tape drives.
- Locates an inventoried file by its name and certain qualifiers that uniquely identify the file.
- Issues tape or disc pack mounting instructions to the operator when necessary.
- Verifies the mounting of labeled volumes.
- Passes descriptive information to the user program opening a file.
- Allows the user of a file to request more storage for the file.
- Denies unauthorized users access to files.
- Returns assigned storage to available storage whenever a file is deleted.
- Maintains a table of contents on each disc volume.

As the largest single component of the ADEPT Eexecutive (65,000 bytes), the Cataloger was written in a new, experimental programming language called MOL-360 (Machine-Oriented Language for the 360).[15] It is a "higher-level machine language" developed under an ARPA-sponsored SDC research project on metacompilers. It resolved the dilemma involving our desire for higher-level source language and our need to achieve flexibility with machine code. The Cataloger design and checkout, enhanced by the use of MOL-360, showed simultaneously the validity of MOL compilers for difficult machine-dependent programming.

The SPAM component

SPAM is a BASEX component that permits symbolic, user-oriented I/O. It can be viewed as a special-purpose compiler that compiles symbolic user program I/O calls into 360 channel programs, and delivers them to the Input/Output Supervisor (IOS) for execution via the EXCP (execute channel program) call. The results of EXCP for the call are "interpreted" by SPAM and returned to the user program as status information. As such, SPAM represents a more symbolic I/O capability than the EXCP level. It provides a relatively simple method for executing the operations of reading, writing, altering, searching for, and positioning records within ADEPT cataloged and controlled disc-based and tape-based file structures.

Resource management

As of this writing, the computer operator has a set of commands at his disposal that allow him to control the system resources. Various privileged on-line commands enable him to monitor the terminal activities of system users and to control assignment and availability of storage devices. However, there is an increasing need for a "manager" to be given more latitude in dynamically controlling the system resources and observing the status of system users, particularly because ADEPT was designed to handle sensitive information in classified government and military facilities. To meet these objectives, a design effort is under way that gives the computer operator system-manager status, with the ability to observe and control the actions of system users. The result will be a program that encompasses some of the management techniques reported by Linde and Chaney[16] tailored to present needs.

Swapping and scheduling user programs

Most of the programs that run under ADEPT occupy all of the core memory that is not used by the resident Basic Executive (46 pages on the 360/50H). If the set of needed pages could be reduced considerable reduction in swap overhead could be expected. One way to achieve this is to mark for swap-out only those pages that were changed during program execution. The hardware needed to automatically mark changed pages is unavailable for the 360/50; however, through use of the store-protect feature on the Model 50, ADEPT software can simulate the effect and produce noteworthy savings in swap time.

Page marking

Whenever a user program is swapped into core, its pages are set in a read-only condition. As the program executes, it periodically attempts to store data (write) in its write-protected pages. The resulting interrupt is fielded by the system. After satisfying itself that the store is legal for the program, the executive marks the target page as "written," turns off write-protect for that page, and resumes the program's execution. The situation repeats for each additional page written. At the completion of the program's time slice, the swapper has a map of all the program pages that were changed (implied in the storage keys with no write protection). Only the changed pages are swapped out of core. Measurement of this scheme shows that about 20 percent of the pages are changed; hence, for every five pages swapped in, only one need be swapped out, for a total swap of six pages, rather than the full swap of ten pages (five in, five out). The scheme makes the drum appear to be 40 percent faster.

The use of the storage protection keys is based on the functional status of each page rather than on some user identity. User programs always run with a program status word key of one, and the bits in the storage key associated with the programs start out at zero. After a page has been initially changed, its key is set to one also. The other bits in the key are used to indicate: first, a page is transient, not yet completely moved to or from swap storage; second, a page is unavailable, i.e., it belongs to someone else; third, a page is locked and cannot be swapped or changed; and finally, a page is fetch-protected because it may contain sensitive information.

Scheduling algorithm

The scheduling algorithm provides for three levels of scheduling. Jobs that are in a "terminal I/O complete" state get first preference in the schedule. Jobs in the second level, or background queue, are run if there are no level-one jobs to run. A job is placed in level two when the two-second quantum clock alarm terminates its operation two consecutive times. Compute and I/O-bound programs are treated alike. A level-two job—when allowed to run—is given quantum interval equal to the basic quantum time multiplied by the scheduling level (i.e., 2 sec × 2 = 4 sec). However, a level-two background job may be preempted after two seconds for terminal I/O. Any operation a level-two job makes that terminates its quantum prematurely will return the job to a level-one status. The batch monitor job is run when the first two queues are empty. User programs may be written to overlap execution and I/O activity. Our choice of scheduling parameters for quantum size, and number of service levels was selected empirically and as a result of prior experience.[17]

A command SKED, which is limited to the operator's terminal, has the effect of forcing top priority for a job (the job stays at level one all the time). Only one job may run in this privileged scheduling state at a time.

Pervasive security controls

Integrated throughout the ADEPT executive are software controls for safeguarding security-sensitive information. The conceptual framework is based upon four "security objects": user, terminal, file, and job. Each of these security objects is formally identified in the system and is also described by a security profile triplet: Authority (e.g., TOP SECRET, SECRET), Need-to-Know Franchise, and Special Category (e.g., EYES ONLY, CRYPTO). At system initialization time, user and terminal security profiles are established by security officers via the system component SYSLOG. SYSLOG also permits the association of up to 64 passwords with each user. At LOGIN time, a user identifies himself by his unique name, up to 12 characters, and enters his private password to authenticate his identity. The LOGIN component of ADEPT validates the user and dynamically derives the security profile for the user's job as a complex function of the user and terminal security profiles. The job security profile is used subsequently as a set of "keys," used when access is made to ADEPT files. The file security profile is the "lock" and is under control of the file subsystem.

File access Need-to-Know is permitted for Private, Semi-Private, and Public use. With the CREATE command, a list of authorized users and the extent of their access authorization (i.e., read-only, write-only, read and write) can be established easily for Semi-Private files. Newly created files are automatically classified with the job's "high water mark" security triplet—a cumulative security profile history of the security of files referenced by the job. Through judicious use of the CHANGE command, these properties may be altered by the owner of the file.

Security controls are also involved in the control of classified memory residue. Software and hardware memory protection is extensively used. Software memory protection is achieved by interpretive, legality checking of memory bounds for I/O buffer transfers, legality checking of device addresses for unauthorized hardware access, and checks of other user program attempts to seduce the operating system into violating security controls.

The hardware protection keys are used to fetch-protect all address space outside the user program and data area. Also, newly allocated space to user programs is zeroed out to avoid classified memory residue.

for that page, and resumes the program's execution. The situation repeats for each additional page written. At the completion of the program's time slice, the swapper has a map of all the program pages that were changed (implied in the storage keys with no write protection). Only the changed pages are swapped out of core. Measurement of this scheme shows that about 20 percent of the pages are changed; hence, for every five pages swapped in, only one need be swapped out, for a total swap of six pages, rather than the full swap of ten pages (five in, five out). The scheme makes the drum appear to be 40 percent faster.

The use of the storage protection keys is based on the functional status of each page rather than on some user identity. User programs always run with a program status word key of one, and the bits in the storage key associated with the programs start out at zero. After a page has been initially changed, its key is set to one also. The other bits in the key are used to indicate: first, a page is transient, not yet completely moved to or from swap storage; second, a page is unavailable, i.e., it belongs to someone else; third, a page is locked and cannot be swapped or changed; and finally, a page is fetch-protected because it may contain sensitive information.

Scheduling algorithm

The scheduling algorithm provides for three levels of scheduling. Jobs that are in a "terminal I/O complete" state get first preference in the schedule. Jobs in the second level, or background queue, are run if there are no level-one jobs to run. A job is placed in level two when the two-second quantum clock alarm terminates its operation two consecutive times. Compute and I/O-bound programs are treated alike. A level-two job—when allowed to run—is given quantum interval equal to the basic quantum time multiplied by the scheduling level (i.e., 2 sec $\times$ 2 = 4 sec). However, a level-two background job may be preempted after two seconds for terminal I/O. Any operation a level-two job makes that terminates its quantum prematurely will return the job to a level-one status. The batch monitor job is run when the first two queues are empty. User programs may be written to overlap execution and I/O activity. Our choice of scheduling parameters for quantum size, and number of service levels was selected empirically and as a result of prior experience.[17]

A command SKED, which is limited to the operator's terminal, has the effect of forcing top priority for a job (the job stays at level one all the time). Only one job may run in this privileged scheduling state at a time.

Pervasive security controls

Integrated throughout the ADEPT executive are software controls for safeguarding security-sensitive information. The conceptual framework is based upon four "security objects": user, terminal, file, and job. Each of these security objects is formally identified in the system and is also described by a security profile triplet: Authority (e.g., TOP SECRET, SECRET), Need-to-Know Franchise, and Special Category (e.g., EYES ONLY, CRYPTO). At system initialization time, user and terminal security profiles are established by security officers via the system component SYSLOG. SYSLOG also permits the association of up to 64 passwords with each user. At LOGIN time, a user identifies himself by his unique name, up to 12 characters, and enters his private password to authenticate his identity. The LOGIN component of ADEPT validates the user and dynamically derives the security profile for the user's job as a complex function of the user and terminal security profiles. The job security profile is used subsequently as a set of "keys," used when access is made to ADEPT files. The file security profile is the "lock" and is under control of the file subsystem.

File access Need-to-Know is permitted for Private, Semi-Private, and Public use. With the CREATE command, a list of authorized users and the extent of their access authorization (i.e., read-only, write-only, read and write) can be established easily for Semi-Private files. Newly created files are automatically classified with the job's "high water mark" security triplet—a cumulative security profile history of the security of files referenced by the job. Through judicious use of the CHANGE command, these properties may be altered by the owner of the file.

Security controls are also involved in the control of classified memory residue. Software and hardware memory protection is extensively used. Software memory protection is achieved by interpretive, legality checking of memory bounds for I/O buffer transfers, legality checking of device addresses for unauthorized hardware access, and checks of other user program attempts to seduce the operating system into violating security controls.

The hardware protection keys are used to fetch-protect all address space outside the user program and data area. Also, newly allocated space to user programs is zeroed out to avoid classified memory residue.

Typically, the complete system reaches "on the air" status in less than a minute.

System instrumentation

Many of the parameters built into the scheduling and swapping of early ADEPT versions were based upon empirical knowledge. The latest versions of the Basic and Extended Executives include routines to record system performance, reliability, and security locks.

Built into the BASEX is a routine to measure the overall and the detailed system performance.[20] Such factors as the number of users, file usage, hardware and software errors, and page transaction response time are recorded on unused portions of the 2303 drum. These measurements provide a better understanding of the system under a variety of inputs and give the designers insight into how the hardware and software components of the system affect the performance of the human user.

An AUDIT program was made part of the EXEX to record the security interaction of terminals, users, and files. AUDIT records EXEX activity in the areas of LOGIN, LOGOUT, and File Manipulation. This routine strengthens the security safeguards of the executive. Specific items that are recorded involve: type of event, user identification, user account number, job security, device identification, time of event, file identification, file security and event success. In addition, this routine provides accounting information and is used as a means of debugging the security locks of new system releases.

In addition to the BASEX recording function, several object programs have been written that simulate various modes of user activity and provide controlled job distributions. These programs, called "benchmarks," run under controlled conditions and enhance the means of improving system performance and throughput, as described elsewhere by Karush.[21] The programs are designed to gather performance measures on the major routines of the executive and have been of considerable help in system "tuning," because they reflect the effect of coding and design changes to various system routines. The routines in the executive that are of primary concern are the swapper, the scheduer, the terminal read/write package, and the interrupt handling processes. Attempts are being made to design a set of benchmarks that represent a typical job mix. However, we are primarily interested in measuring the performance of our system against various modifications of itself and in measuring its behavior with respect to different job mixes.

SUMMARY

The ADEPT executive is a second-generation, general-purpose, time-sharing system designed for IBM 360 computers. Unlike the monolithic systems of the past,[1,2] it is structured in modular fashion, employing distributed executive design techniques that have permitted evolutionary development. This design has not only produced a flexible executive system but has given the user the same facilities used by the executive for controlling the behavior of his programs. ADEPT's security aspects are unique in the industry, and the testing and fabrication methods employ a number of novel approaches to system checkout that contribute to its operational reliability.

It is important to note that this system deals particularly well with size limitation problems of very large files and very large programs. The provisions made for multiple programs per job, active/inactive page status for programs larger than core size, page sharing between programs, common file access across programs within jobs, and the commitment of considerable space to active file environment tables (up to four pages worth) contribute to this success. Nevertheless, all these capabilities are designed to handle the smaller entities as well. We feel ADEPT-50 is a significant contribution to the technology of general-purpose time-sharing.

ACKNOWLEDGMENTS

We would like to express our appreciation for the dedicated efforts of some very *adept* individuals who participated in the design and building of this time-sharing system. Our thanks go to Mr. Salvador Aranda, Mr. Peter Baker, Mrs. Martha Bleier, Mr. Arnold Karush, Mrs. Patricia Kribs, Mr. Reginald Martin, Mr. Alexander Tschekaloff and all the others who have followed their lead.

REFERENCES

1 P CRISMAN editor
The compatible time-sharing system: A programmer's guide
MIT Press Cambridge Mass 1965

2 J SCHWARTZ et al
A general-purpose time-sharing system
Proc SJCC Vol 25 1964 397-411 Spartan Books Baltimore

3 E W FRANKS
A data management system for time-shared file-processing using a cross-index file and self-defining entries
AFIPS Proc Vol 28 1966 79-86 Also available as SDC document SP-2248 21 April 1966

4 R E BLEIER
Treating hierarchical data structures in the SDC time-shared data management system (TDMS)
Proc 22nd Nat ACM Conf Thompson Book Co 1967 41-49

5 E W DIJKSTRA
The structure of T.H.E. multi-programming system
C A C M Vol 11 No 5 May 1968

6 F J CORBATO V A VYSSOTSKY
Introduction and overview of the multics system
Proc FJCC Nov 30 1965 Las Vegas Nevada

7 B W LAMPSON
Time-sharing system reference manual
Working Doc Univ of Calif Doc No 30.1030
Sept 1965 Dec 1965

8 B W LAMPSON
A scheduling philosophy for multi-processing systems
C A C M Vol 11 No 5 May 1968

9 J H SALTZER
Traffic control in a multiplexed computer system
MAC-TR-30 thesis MIT Press July 1966

10 G H FINE et al
Dynamic program behavior under paging
Proc ACM 1966 223-228 Thompson Book Co Wash D C

11 E G COFFMAN L C VARIAN
Further experimental data on the behavior of programs in a paging environment
C A C M Vol 11 No 7 July 1968 471-474

12 L A BELADY
A study of replacement algorithms for a virtual storage computer
IBM Systems Journal Vol 5 No 2 1966

13 R W O'NEIL
Experience using a time-shared multi-programing system with dynamic address relocation hardware
Proc SJCC 1967 Vol 30 611-627 Thompson Book Co Washington D C

14 L G ROBERTS
Multiple computer networks and intercomputer networks and intercomputer communication
ACM Symposium on Operating System Principles
Oct 1-4 1967 Gatlinburg Tenn

15 E BOOK D C SCHORRE S J SHERMAN
Users manual for MOL-360
SCC Doc TM-3086/003/01

16 R R LINDE P E CHANEY
Operational management of time-sharing systems
Proc ACM 1966 149-159

17 P V McISSAC
Job descriptions and scheduling in the SDC Q-32 time-sharing system
SDC Doc TM-2996 June 1966 28

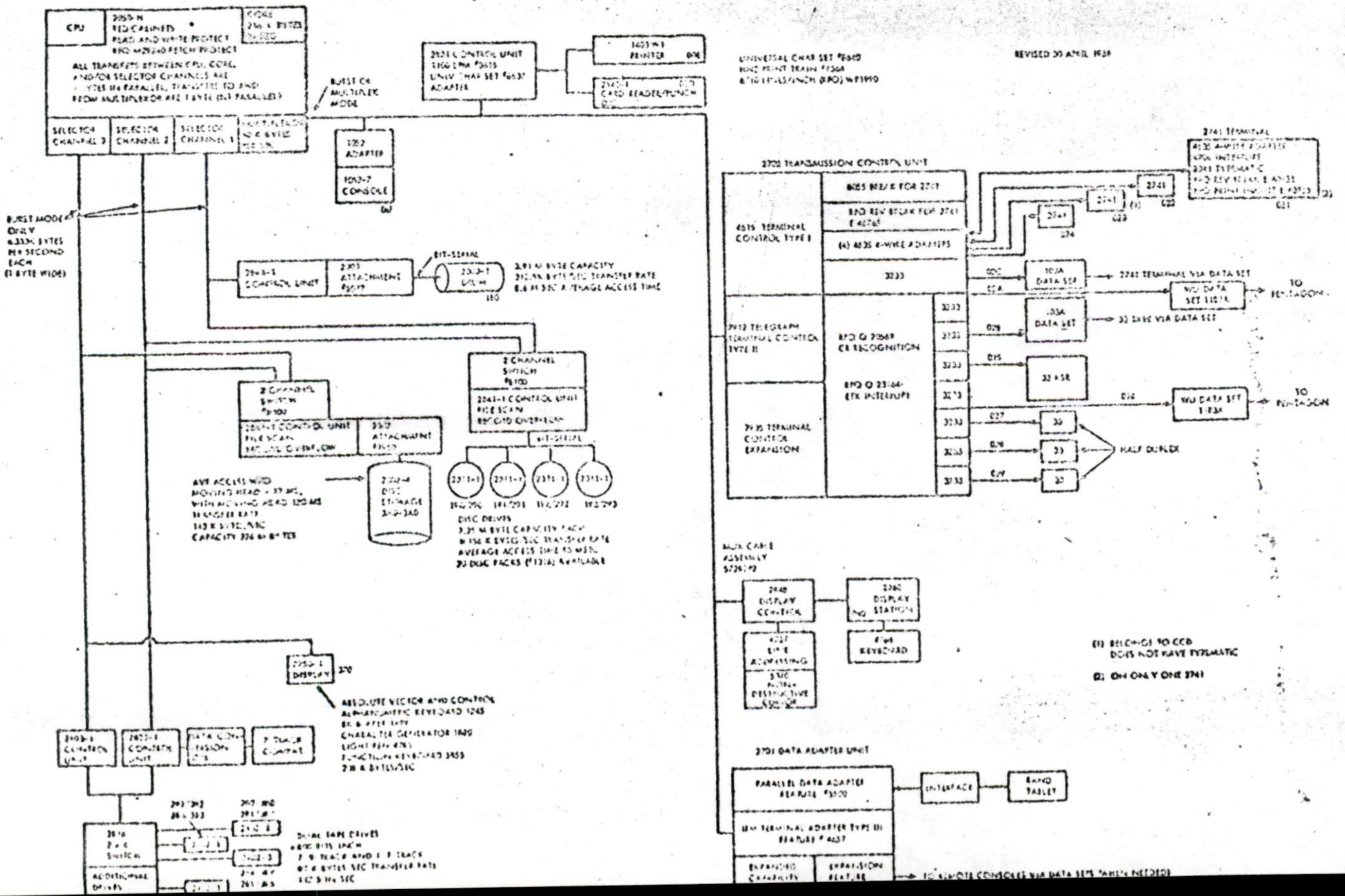
18 C WEISSMAN
Security controls in the ADEPT-50 time-sharing system
AFIPS Proc FJCC Vol 35 1969

19 W A BERNSTEIN J T OWENS
Debugging in a time-sharing environment
AFIPS Proc FJCC Vol 33 1968 7-14

20 A D KARUSH
The computer system recording utility: application and theory
SDC Doc SP-3303 Feb 1969

21 A D KARUSH
Benchmark analysis of time-sharing system
SDC Doc SP-3343 April 1969

APPENDIX A: Advanced development prototype system block diagram.



SIC Report

NATIONAL ACADEMY OF SCIENCES

COMPUTER SCIENCE & ENGINEERING BOARD
2101 CONSTITUTION AVENUE
WASHINGTON, D. C. 20418

June 1, 1970

Mr. Milo Peterson
Chairman, Technical Committee on
Industrial Classification
Bureau of the Budget
Washington, D. C. 20503

Dear Mr. Peterson:

The Data Base Panel of the Computer Science and Engineering Board has been examining in depth the dynamics and economics of the Computer Industry in order to take advantage of the present opportunity for revision of the Standard Industrial Classification. Of particular interest to this panel is the significance of the contribution of the total information processing industry to our economy. Of particular concern to this panel is the lack of definitive data as to what is being produced by this industry and what trends prevail.

We are informally submitting herewith a suggested SIC format to adequately define the Information Processing Equipment and Supplies as well as the Information Processing Product and Services activities in the U. S. with (where available) 1969 estimates of activity and 1974 projections. It is our opinion that the total significance of the contribution and activity of the Information Processing Industry and its potential growth justifies it receiving 2 two-digit SIC designators, one for Equipment and Supplies, the other for Product and Services. In the attached, we have identified these as 3X, the X standing for a second digit which is convenient to the Bureau's planning, and 74 respectively.

In support of this suggestion, we offer the following evaluation of category 3X to indicate its economic significance. Published 1967 Census data has been used for this calculation despite the fact these figures are smaller than the 1969 estimates shown in the suggestion and far short of 1964 projections. Applying the formula (see Table 1) for manufacturing categories to the data in the 1967 Census for SIC 3573 (this includes 3X1 and 3X2 of our suggestion) results in a significance factor of 313%. The 1967 Census figures also indicate specialization and coverage factors of 94% each. If, in addition to 3X1 and 3X2, we add the category 3X3 (Adding Machines, Desk Calculators, Accounting Machines, and Tabulating Machines), the economic significance factor goes to 412%. Continuing

Mr. Milo Peterson
1 June 1970
Page 2

this sequence by the addition of 3X4 (Typewriters) brings the factor to 483%. Of course the specialization and coverage ratios decrease with this addition, but not too greatly. We estimate these would become approximately 85% each.

These figures can be compared to those obtained for other SIC two-digit categories using the same formula and the corresponding 1967 Census data. SIC 21 is calculated to be 270% and SIC 31 660%. These industries (Tobacco and Leather) do not exhibit the dynamic characteristics of the Information Processing Industry and hence we believe that the proposed SIC 3X will easily surpass SIC 31 in a few years. Within the next ten years, SIC 3X may compare very favorably with other manufacturing groups.

We would welcome the opportunity of informally discussing our suggestion with you or the Technical Committee on Industrial Classification at anytime and assure you of any co-operation we are capable of providing.

Sincerely yours,

[illegible]

S. Fernbach
Chairman, Data Base Panel

SF/adc

Enclosure

TABLE I

The formula used is that offered by BOB, namely

$$1/8\left[\frac{\text{No. of Establishments}}{715} + 2\,\frac{\text{No. of Employees}}{38,000} + 2\,\frac{\text{Payroll}}{\$219,000} + 2\,\frac{\text{Value added}}{\$448,000,000} + \frac{\text{Value of Shipment}}{\$998,000,000}\right]$$

The 1967 Census date is summarized below

	No. of Establishments	No. of Employees (thousands)	Payroll (thousands)	Value added (millions)	Value of Shipments (millions)
Computers and peripheral Equipment (Current SIC 3573)	175	98	$798	$1,921	$3,761
Above plus Accounting machines (Current SIC 3573 & SIC 3574)	313	136	$1,092	$2,439	$4,469
Above plus Typewriters (Current SIC 3573 + SIC 3574 + SIC 3522)	338	163	$1,265	$2,894	$5,065

The Data Base Panel

of the

Computer Science and Engineering Board

of the

National Academy of Sciences

PROPOSED CHANGES TO THE STANDARD INDUSTRIAL CLASSIFICATION

ABOUT THE SIC

The U. S. Standard Industrial Classification (SIC)* is used by all Government agencies (and by many private concerns, such as Dun and Bradstreet) as a common framework for collecting and reporting industrial, commercial, and economic statistics. The SIC provides a four-digit industry code for identifying the major economic activity of any "establishment," which, to simplify data reporting and collecting, is vaguely defined and need not necessarily correspond to any organizational or geographic entity.

Related industries are organized into groups, which are distinguished by the first three digits of their SIC code. Related groups are organized into major groups, distinguished by the first two digits of the code, and related major groups are organized into the following divisions: (a) agriculture, forestry, and fisheries; (b) mining; (c) contract construction; (d) manufacturing; (e) transportation, communication, and utilities; (f) wholesale and retail trade; (g) finance, insurance, and real estate; (h) services; (i) government; (j) nonclassifiable establishments.

Since the digit 0 is used to identify an unknown establishment (where not enough information is available to completely classify it), and since the digit 9 is used to identify miscellaneous categories, a group may contain no more than eight, non-miscellaneous industries, and a major group no more than eight non-miscellaneous groups.

An inter-agency Technical Committee on Industrial Classification, chaired by Milo Peterson of the Bureau of the Budget, is now engaged in revising the SIC. They will consider proposed revisions from any source, if accompanied by documented justification, until June 30, 1970. The revised edition of the classification will be effective January 1, 1972.

* Published in the "Standard Industrial Classification Manual, 1967," available from the Government Printing Office for $4.50 a copy.

According to the criteria established by this Committee, a proposed new industry should be significant, specialized, and inclusive. A new industry is considered to be significant if it has at least 20 percent of the number of establishments, employees, and dollar-value-added of the average industry in the same division. It is considered to be specialized if at least 80 percent of the products produced by establishments in the industry are those defining the industry. And it is considered to be inclusive if at least 70 percent of the products defining the industry (50 percent where these products are produced in significant amounts in other industries for internal use) are produced by establishments in the industry.

Still another criterion is comparability. A new classification that involves only the simple amalgamation or division of current industries facilitates comparisons with statistics kept under the old classification, and is thus preferred to a classification wherein new industries are made up of bits and pieces of several current industries. The fragmentation of miscellaneous industries and groups is presumably acceptable, however.

COMPUTING ITEMS IN CURRENT SIC MANUAL

Establishments in what is commonly called the computing industry are currently identified by the following codes:

2645	Die Cut Paper and Paperboard and cardboard (which includes tabulating card manufacturing)
3573	Electronic Computing Equipment
7392	Business, Management, Administrative and Consulting Services (which includes computer programming services)

7394	Equipment Rental and Leasing Services (which includes electronic equipment rental and leasing)
8242	Vocational Schools (which includes data processing schools)
8931	Accounting, Auditing, and Bookkeeping Services (which includes data processing services)

The Panel proposes that the Board recommend changes to the SIC that will better reflect both the structure and the importance of the computing industry as it now exists and as it is likely in the next few years to become.

The Panel's goals in this are two-fold: (1) that the revised SIC permit the separate identification of all the various sub-industries that make up the computing industry and (2) that it reflect the present and near future structure of the computing industry in as much detail as is consistent with the established criteria.

The Panel's proposed change to the SIC calls for the segments of the computing industry, with a few necessary exceptions, to be concentrated in two Major Groups -- one, Information Processing Equipment and Supplies, within the Manufacturing Division, (which would replace Group No. 357: Office, Computing, and Accounting Machines), the other, Information Processing Products and Services, a new Major Group within the Services Division.

The data in support of the proposal was prepared by the International Data Corporation at the request of the panel. The following comments by IDC indicate the significance of the data.

The estimates for 1969 are based on an extensive survey and analysis of the expenditures for computer-related products and services by users of computers and data processing equipment in the United States. We have estimated exports of computer-related equipment and services in order to provide an estimate of the total value of equipment and services produced by establishments within the United States. The preliminary estimates were checked against statistics on the computer installation census file maintained by IDC, from various trade reports, government statistics, and related information.

The counts for the number of establishments were estimated from an analysis of trade directories, association officers, annual reports and related references. There establishment was defined as a single physical location engaged in the identified activity. Generally it was a plant, office, or local service facility. Particularly in the case of the equipment manufacturing sectors, each of the principal firms engaged in the designated industry have several to over 50 establishments.

The "number of employees" estimate was prepared from an analysis of questionnaires received from firms designated in industry, trade directories, annual reports, government statistics and related information. An employee was counted as participating in an industry if he spent at least 25% of his time engaged in the activities of that industry. There is, therefore, considerable double-counting of employees in the computer systems/peripheral equipment sector where administrative and marketing personnel support the administration and sales of heterogeneous product lines.

Likewise, in the proprietary software/programming/systems analysis sector, many of the management, marketing, and technical personnel are involved simultaneously with two or three of these activities on a continuing basis.

The forecasts for 1974 are made by combining projections of customer demands established by customer requirement studies, investigation of the untapped potential of the user computer systems, analysis of the contributions of new information technology to the growth of computer applications, a projection of the economic environment under which the sales of computer related products and services will take place. Because of the high levels of uncertainty involved in each of the elements of this projection, the figures must be considered highly speculative. This is particularly true in the measurement of the number of establishments.

oposed		Proposed Industry Description	Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports)					
			Size Measures					
			1969			1974		
			No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)	No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)
INFORMATION PROCESSING EQUIPMENT AND SUPPLIES								
3X1	COMPUTERS							
	3X11	Digital Computers (Value relates to Central Processor and internal core storage only)	375	148,000	2260	350-420	160,000	3300
	3X19	Miscellaneous Computers, including Analog and Hybrid Computers (not elsewhere classified)						
		Estimates for Analog computers	40	2,800	52	35-45	1,350	30
3X2	COMPUTER PERIPHERAL EQUIPMENT							
	3X21	Punched Card Handling Equipment (on and off-line)	30	100,000	700	40-50	150,000	920
	3X22	Printers	40	95,000	650	50-60	105,000	885
	3X23	Optical and Magnetic Character Readers and Writers	35	20,000	110	55-65	50,000	260
	3X24	Display Equipment (including graphic display, plotters, and interactive line and character displays)	25	18,000	40	70-85	50,000	165

roposed C o.	Proposed Industry Description	Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports) — Size Measures — 1969			1974		
		No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)	No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)
3X25	Auxiliary Storage Equipment (magnetic tape drives, disk files, disk pack drives, magnetic card and strip readers, magnetic drums, and related equipment.)	45	130,000	1650	68-85	155,000	2600
3X26	Computer Terminals (Including Conversational and Remote Batch, except purely communication terminals)	140	96,000	220	250-300	120,000	960
3X29	Miscellaneous						
3X3	ADDING MACHINES, DESK CALCULATORS, ACCOUNTING MACHINES AND TABULATING MACHINES (Now listed as 3574)						
3X4	TYPEWRITERS (Now listed as 3572)						
3X5	COPYING AND DUPLICATING MACHINES (Now listed under 3579)						
3X6	MICROFORM EQUIPMENT						
3X7	INFORMATION PROCESSING EQUIPMENT SUPPLIES AND ACCESSORIES	450	48,000	880	500-600	70,000	1410

oposed	Proposed Industry Description	Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports)					
		Size Measures					
		1969			1974		
		No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)	No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)
3X9	MISCELLANEOUS INFORMATION PROCESSING EQUIPMENT						
3X91	Mailing and Addressing Equipment (Now listed as 3579)						
3X92	Scales and Balances, Except Laboratory (Now listed as 3574)						
3X99	Information Processing and Office Equipment, Not Elsewhere Classified.						

roposed IC o.	Proposed Industry Description	1969 No. of Estab.	1969 Total Employ.	1969 Value of Shipments or Services ($ Mill.)	1974 No. of Estab.	1974 Total Employ.	1974 Value of Shipments or Services ($ Mill.)
4	INFORMATION PROCESSING PRODUCTS AND SERVICES						
741	COMPUTER PROGRAMS AND PROGRAMMING SERVICES						
7411	Proprietary Computer Programs (Computer programs designed to be sold with no or a minimum amount of customization to the individual customer.)	200	2,500	20	700-900	100,000	1550
7412	Computer Programming Services (custom contract services in programming and coding computer instructions, including work done on an hourly or daily rate basis.)	1000	21,000	360	2000-3000	110,000	1335
7413	Systems Analysis, Design, Evaluation, Selection and Consulting Services (Limited to those directly involved with the application of computer systems.)	300	3,800	65	1500-2000	110,000	625
7419	Computer Programs and Programming Services (Not Elsewhere Classified)						

Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports) — Size Measures

Estimated Value of Shipments of Services
Within Establishments in the United States
(including exports)

Size Measures

posed	Proposed Industry Description	1969			1974		
		No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)	No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)
742	COMPUTATIONAL AND ALLIED SERVICES						
	7421 Computing Services	2500	28,000	860	3800-5000	130.000	1800
	7422 Data Preparation and Conversion Services	2800	12,000	80	3000-4000	28,000	175
	7423 Computer Facility Management and Turnkey System Development	30	1,200	20	200-300	8,000	100
	7429 Miscellaneous Computational Services (Not Elsewhere Classified)						
743	COMPUTER-BASED TEXTUAL SERVICES						
	The preparation and/or marketing of documentary, bibliographic, abstracting, indexing, catalog, search and retrieval products and services which rely significantly on computer manipulation, preparation, or storage of the informational material.	500	10,000	400	1,000	20,000	800

oposed C .	Proposed Industry Description	Estimated Value of Shipments or Services Produced Within Establishments in the United States (including exports)					
		Size Measures					
		1969			1974		
		No. of Estab.	Total Employ.	Value of Shipments or Services ($ Mill.)	No. of Estab.	Total Employ.	Value of Shipments or Service ($ Mill.)
744	INFORMATION PROCESSING EQUIPMENT REPAIR AND MAINTENANCE	400	40,000	500	700-800	75,000	1100
745	INFORMATION PROCESSING EQUIPMENT RENTAL AND LEASING AND USED EQUIPMENT SALES (OTHER THAN THE MANUFACTURER)	250	5,000	330	350-400	10,000	700
746	ACCOUNTING, AUDITING, AND BOOKKEEPING SERVICES (replaces current Group No. 893)						
749	MISCELLANEOUS INFORMATION PROCESSING PRODUCTS AND SERVICES (NOT ELSEWHERE CLASSIFIED)						

In addition to these two major groups, the Panel recommends one new category; namely,

7362 Information Processing Employment Agencies

and the addition of,

Data Processing Vocational Schools
(under Group No. 8242)

within Group No. 736 (Private Employment Agencies) and Group No. 824 (Vocational Schools, Except Vocational High Schools) respectively.

Auerbach
Proposal

information and management sciences

AUERBACH
121 n. broad st.
philadelphia
penna. 19107
215-491-8200
cable: auerinfo

June 12, 1970

Dr. Anthony G. Oettinger, Chairman
Computer Science & Engineering Board
National Academy of Sciences
2101 Constitution Avenue
Washington, D. C. 20418

Subject: Proposal for International Computer Activities Panel

Dear Tony:

This is in response to your request to formulate a proposal for the establishment of an International Computer Activities Panel to monitor and report to the Computer Science & Engineering Board on subjects of possible interest in the international arena of computer sciences and engineering.

Mission:

1. To assess the importance of international computer sciences and engineering activities to the industry and the country and to report on them periodically

2. To report on matters of significance pertaining to the activities of intergovernmental organizations in the computer sciences and engineering field.

3. To provide advice, guidance and methods for obtaining assistance to the Foreign Secretary of the National Academy of Sciences pertaining to international matters in the field of computer sciences and engineering and relevant to requests from AID, OECD, and similar organizations and for tours requested from foreign delegations.

philadelphia
washington
new york
boston
san francisco
london

AUERBACH

Plan of Action:

The initial efforts of the Panel will be to segment the international computer sciences and engineering activities so that task forces may be assigned to assess and report on the importance of overseas developments to our country.

A survey will be assembled on significant intergovernmental activities in the computer sciences and engineering field, for the Board.

Finances:

An actual budget of $20,000 for the formative stages is requested. Additional grants will be requested as the work becomes more specific.

Suggested Panel Members:

It is proposed that initially the Panel consist of six individuals who are knowledgeable in the international computer sciences and engineering field. The Panel will be expanded as the missions are better defined.

The suggested candidates are:

Dr. Willis H. Ware
The RAND Corporation
1700 Main Street
Santa Monica, California 90406
213 393 0411

Dr. Richard Tanaka, Vice President
California Computer Products
305 N. Muller Street
Anaheim, California 92803
714 774 9141

Mr. Benjamin Kessel, Vice President
International Computer & Communications Division
Honeywell, Inc.
60 Walnut Street
Wellesley Hills, Mass. 02181
617 235 7450

AUERBACH

Dr. Herbert Freeman
50 Shelley Lane
Great Neck, New York 11023
516 482 7748

Dr. William F. Atchison, Director
Computer Science Center
University of Maryland
College Park, Maryland 20742

Mr. Harry D. Huskey
Computer Center
University of California
Santa Cruz, California 95060
408 429 0111

Mr. Hugh P. Donaghue
Control Data Corporation
Facility Code WSAASO
2000 L Street, N. W. - Suite 424
Washington, D. C. 20036
202 296 0200

Representative from General Electric Company

Representative from International Business Machines Corporation

Sincerely,

Isaac

Isaac L. Auerbach
President

Knox-Baer Prop

Xerox to Knox + all these copied.

9:00 to

MAY 25 REC'D

McGRAW-HILL, INC.

330 WEST 42ND STREET, NEW YORK, N.Y. 10036

WILLIAM T. KNOX
VICE PRESIDENT

(212) 971-6486

May 11, 1970

Dear Tony:

Wally's April 20th letter emphasizing the opportunity for retraining physicists in computer science makes a very good point, and I, too, would like to see something done about it. All the public and private programs for graduate training that I know of are predicated on the assumption that the student is pursuing at the graduate level a specialty for which he has received undergraduate training.

Wally suggests a small scale, experimental Federal support program might be started. I would prefer to see the CSEB push for a privately supported program first. Some of the larger private foundations, many of which have long been interested in education might find a retreading program quite interesting. If the program is successful under private auspices, then there is some reason to hope that it will work under Federal auspices. Starting such a program under Federal auspices might doom it either to failure or gross inefficiencies from the very beginning, and give the whole idea of retreading a black eye.

Sincerely,

Bill

William T. Knox

Dr. Anthony G. Oettinger
Harvard University
Cambridge, Massachusetts 02138

cc: Messrs. W. S. Baer
W. C. House
A. J. Perlis
J. R. Pierce

A great idea! Could you and Baer draft a proposal and suggest who we should take it to? It would be nice to have that ready for [illegible] T. [illegible] 17-18) BAER: 213-277-2900

Beiser-CAD/CAM

PRESENTATION PAPER ON CAD/CAM

The increasing interest of industry and government agencies -- particularly the Department of Defense -- in Computer Aided Design and Computer Aided Manufacturing (CAD/CAM) has reached a point where a central authoritative office is needed to serve as a national coordinating center. Specifically, a group of individuals from industry that has been working on a wide variety of CAD/CAM problems would like to explore possible affiliation with the Computer Research & Engineering Board to continue their work on such topics as new technology, education and management in this subject area.

The background on the above development involves the gathering of a group of specialists to assist the Department of Defense in putting on the national CAD/CAM conference that was held at Davenport, Iowa, during October of 1969. This group was organized into panels, 12 of which covered different aspects of CAD/CAM operations. Panel 13 summarized the findings of the twelve panels and Panel 14 assumed the responsibility for preparing recommendations that would benefit the national CAD/CAM program. One of these recommendations expresses the need for a national coordinating center.

Panel 14 is the only panel that is still active. It is made up of two representatives from each of the following engineering societies and associations: National Security Industrial Association, American Ordnance Association, Electronic Industries Association, Aerospace Industries Association, the Society of Manufacturing Engineers, and the Numerical Control Society. This group has worked well together for many months and has been quite effective in achieving its immediate objectives. It now feels the need for communicating and planning with a knowledgeable permanent group on a series of problems which require further attention. These problems exist in areas such as training, languages, standards, contracting, etc.

Panel 14 will meet on 1 & 2 of May at Cocoa Beach, Florida, to further consider selected problems that it believes warrant immediate attention. If your reaction to this inquiry is favorable, plans could be made for more formal discussions and presentations at a future meeting of the Board.

Geo Berser

Brazil

9:00 W
JUN 15 REC'D

NATIONAL ACADEMY OF SCIENCES
2101 CONSTITUTION AVENUE
WASHINGTON, D.C. 20418

OFFICE OF THE FOREIGN SECRETARY

June 12, 1970

Mr. Warren House
Executive Secretary
Computer Science and Engineering Board
National Academy of Sciences

Dear Mr. House:

On behalf of the Foreign Secretary, I want to thank you for the considerable help you gave our Office in regard to the visit of Professors Denis Leite and Antonio Olinto, the Brazilian computer scientists. The advice on universities to visit and people to meet was particularly valuable. Because of this guidance, the Brazilian scientists were extremely well received at each institution.

This success has permitted the Office of the Foreign Secretary to move ahead in the Brazil-U.S. computer science program. Four U.S. computer scientists have been nominated to serve on the U.S. panel and each has accepted. Significantly, the four gentlemen were selected from among the computer scientists met by Professors Leite and Olinto during their trip, and three have some prior experience in Brazil. The four professors are:

1. Harry D. Huskey, Chairman of the U.S. Panel, University of California, Santa Cruz, California.

2. Barry W. Boehm, Mathematician, Rand Corporation, Santa Monica, California.

3. Bruce Gilchrist, Executive Director, American Federation of Data Processing Societies (AFIPS), Montvale, New Jersey

4. Michel A. Melkanoff, Chairman, Dept. Computer Science, University of California, Los Angeles, California.

The status of the program is that these names have just been submitted to the Brazilian National Research Council, along with the suggestion for the first meeting of the joint study group in Rio during the week of August 10-15, 1970.

Again, I thank you and Mr. Kettler for your advice.

Sincerely yours,

James G Zavistoski

James G. Zavistoski
Professional Associate
Board on Science and Technology
for International Development