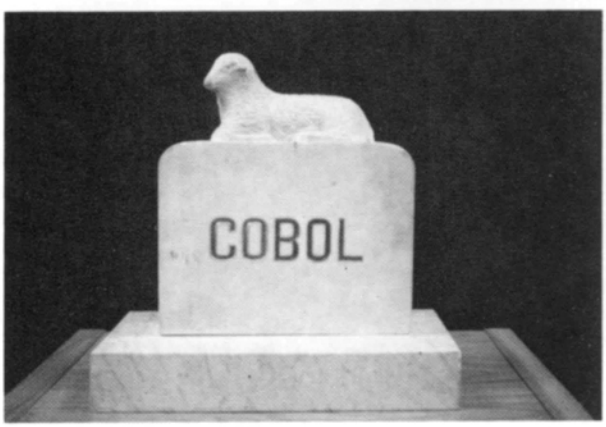
A personal recollection of the development of the COBOL programming language, complementing the formal histories. With this background, some possibilities and recommendations for the future of COBOL are given.

a view of The History of COBOL

R. W. Bemer



At a time when the future of COBOL was less certain, Charlie Phillips received a package from Howard Bromberg, express collect. This is the tombstone that he paid $15 for.

FROM THE EDITOR

This article first appeared in M. Berk's "The Programmer's COBOL", Inter-ACT/McGraw-Hill. It is reprinted with permission of the publishers for the large number of people who use the COBOL language without any real idea of how it came to be. A historical framework, if not too dull, can add some interest to the daily work.

The content has been verified by many of the active participants. The article attempts to give full credit to the various organizations that participated. Although the viewpoint (not bias, hopefully) is necessarily that of an IBM employee at the time, the author is now with Honeywell Information Systems, and can take much pride in their part of the development of COBOL.

It isn't all history, however. There are some predictions for the future. Some of these will come to pass without user pressures; some will not. Perhaps readers will see here some viewpoints and standards actions that they should be supporting for their own best interests, and for the best interests of the computer-using world.

COBOL is a programming language known to a large number of people who work in data processing. It is less known to the general public, and chemists becoming involved in data processing for the first time are prone to confuse it with Element No. 27. It is unique among the major programming languages in that factual histories of its inception and development abound. Standard summaries of the history are carried in all official documents and in the manuals for specific machine implementations, a policy of the original sponsors.

A very complete history and summary of activities is to be found in the publication "American National Standard X3.23, COBOL". With this, one can track meetings, participating personnel, and technical motivation. Omitted are elements of personality, background, competition, infighting, and significance to the data processing world. Reading this history and others, one might conclude that there was never any excitement, strategy, or corporate and individual struggles. Not so.

The very fact that these standard histories are carried in official COBOL documents is a key to understanding the COBOL effort. COBOL is intended to conserve costs and human resources, but any of the proprietary languages of its class could have done that, and very possibly they would all have grown and matured in the same way. IBM's FORTRAN became an industry standard because it was operational in volume before its competitors, and because IBM placed it in the public domain. In the business data processing world, the race was much closer.

ORIGINS

In time sequence of development, the three progenitors of COBOL were: FLOW-MATIC (from UNIVAC), Commercial Translator (from IBM, which ran into legal conflicts with the original name COMTRAN), and AIMACO (from the Air Materiel Command in Dayton).

FLOW-MATIC was an outgrowth of the A-series of algebraic and scientific compilers. The concept of the compiler is largely due to Dr. Grace Murray Hopper, who was in charge of these projects. The new series started as B-0 (B for business, as opposed to A for algebraic). A predecessor, BIOR (Business Input Output Rerun), was developed by a different group, becoming somewhat operational in 1955 April. FLOW-MATIC, as B-0 was renamed, became operational for the UNIVAC I and II in 1956 December. It was not what we would call today a commercial-grade software product, and both language and compiler were still undergoing continual change and improvement.

The competitive threat potential of FLOW-MATIC did not go unnoticed at IBM, and some research was started in the Fall of 1956 on alternate solutions for a business language. The original approach tended to high-level operations and set notation, such as "MERGE FILEA WITH FILEB ON KEY 3" and "UPDATE THIS WITH THAT". I began to worry that this approach might take too long to bring to practicality, and asked Roy Goldfinger to develop a language with the more specific procedural capability of FLOW-MATIC, yet which would retain the set principles. Public notice of the Commercial Translator work was given to the SHARE group in 1957 October, and Roy produced formal specifications in 1958 March.

Reading the standard COBOL histories, one could get an impression that the early meetings were spontaneous. Actually, Mary Hawes of Burroughs had buttonholed Dr. Saul Gorn of the University of Pennsylvania at the Western Joint Computer Conference in San Francisco on 1959 March 3-5, asking if he didn't think it was time for a common business language. Saul agreed and later that month held a meeting in his office at the (UNIVAC I) Computer Center. At a second meeting on April 8, various names were suggested for leadership. Grace Hopper proposed Charles Phillips of the Department of Defense. It seemed most reasonable for DOD to sponsor such an effort, which would need energetic leadership and neutrality, together with the stature (and pocketbook) to command the attention of the manufacturers. I sent the agenda for the May 28 meeting to

Some members of the CODASYL Executive Committee witnessing the compatibility demonstration on 1960 December 6. From left to right — Smith, Dillon, Albertson, (Danny Goldstein), Phillips, Grosz, (Jack Jones), Curry.

the President of the Association for Computing Machinery (probably as a needle for the disdain of business languages in the ACM Programming Languages Committee, although they were within their scope to develop).

THE PEOPLE

The Washington meeting of May 28 noted the separate development of three similar languages, and agreed that the example of ALGOL warranted an effort to develop a common business language. A steering (later executive) committee was formed with Phillips as Chairman, Joe Cunningham of the Air Force as Vice Chairman, Gene Albertson of U.S. Steel, Greg Dillon of DuPont, Mel Grosz of ESSO, plus the chairmen of the three task groups formed. They were: the Short Range, under Joe Wegstein (one of the founders of ALGOL) of the National Bureau of Standards; the Intermediate Range, under Gene Smith of the Bureau of Ships; and the Long Range, under Bob Curry of Southern Railway. I name these people here because they were all active for a long time in the COBOL world, and it is doubtful if history would have been the same without their efforts. Grace Hopper and I were appointed as technical advisors. My nontechnical contribution was the coining of CODASYL, for the Conference on Data Systems Languages. We can't find a single individual who admits coining the acronym "COBOL".

EARLY RESULTS

Some of the old hands laughed disbelievingly when the Short Range task group was charged to come up with a composite within three months. Still, they worked assiduously, and held twelve meetings between June 23 and the end of August. Membership included representatives of Burroughs, IBM, Minneapolis Honeywell, RCA, Remington Rand UNIVAC, and Sylvania, plus the Air Materiel Command and the Bureau of Ships. The report was presented to the Executive Committee on September 4. Wegstein said charitably that "It contains rough spots and requires some additions". It was, in fact, a committee hodgepodge, which hardly caused IBM to abandon plans for Commercial Translator. The task group was enjoined to get it in shape by December 1, and to continue in existence beyond that to monitor implementations on various computers.

COMPLICATIONS

Now uncertainty enters. A news item in June had stated that a new company was being formed (Computer Sciences Corporation) whose first responsibility would be the construction of a compiler for the Honeywell 800. Joe Wegstein gave details to the Executive Committee on September 4, when he presented his report. Unquestionably the Intermediate Range Committee (the task groups had by now been upgraded) was dismayed by the first results of the Short Range Committee. One of its members was Dr. Richard Clippinger of Honeywell, who was able to furnish a copy of the specifications for the Honeywell Business Compiler Language (later FACT) at their October 8 meeting. FACT was a medley of the three approaches tried for Commercial Translator: (1) modular report, file maintenance and sort generators, (2) high-level operators such as "UPDATE", and (3) the English language procedural statements. As a language, it was undoubtedly rich and well-defined for that period; one could well say that it was ahead of its time.

FACT was too attractive to the Intermediate Range Committee in the face of the shortcomings of the first COBOL report; they endorsed FACT to be the basis of the common business language. The word spread with shock waves. IBM and UNIVAC, having consented to work on a composite, could scarcely be happy to scrap the work and accept *in toto* the language of a competitor. General doubt was expressed about the sanity of the Intermediate Range Committee, which retaliated by convening again just five days later and reaffirming its support of FACT: 15 in favor, 1 opposed, and 2 abstaining.

Unfortunately for FACT, the implementation failed the language. Bob McDowell (first with Computer Sciences Corporation, and later Honeywell's liaison with them) says that this was caused primarily by an attempt to force the compiler into too small a configuration, resulting in missed schedules and customer problems (unlike 3rd-generation customers, they were not resigned to such difficulties). He feels that if FACT had been at least partially demonstrable at the time, the battle may have been won. However, in view of the competitive situation, the war might have been lost.

As it turned out, FACT was not operational until a year after the first COBOL processors. Nevertheless, the language was a substantial contributor to COBOL, and

this is reflected in the later acknowledgments. In contrast, AIMACO as a language was not; it was a derivation of B-0 undertaken as a joint project with UNIVAC, and the contribution here was from the personnel of the Air Materiel Command. The Short Range Committee submitted its next report in 1960 January. It was accepted, subject to editing for "typographical and other minor errors". The editing committee was chaired by Phillips, together with Wegstein and Betty Holberton. The minor work took from January through April, and the COBOL 60 Report was issued by the Government Printing Office in June.

COBOL 60

Despite the inadequacies of the Report, the list of manufacturers announcing or committing to COBOL implementations grew. It was simply the thing to do. However, many qualms were felt about a language defined in large part by example rather than by syntax and semantics, particularly at IBM, whose John Backus had presented his metalinguistic notation (BNF) the previous summer. General Electric's Charlie Katz, another old ALGOLer, warned (in the public announcement of their language GECOM) that while COBOL could be accepted by GECOM, it was not yet developed to the point where unambiguous interpretation was possible.

The official IBM position on COBOL was a critical element for acceptance in the industry. Commercial Translator had been announced for the 7070, 709/90, and 705 III. Barry Gordon was responsible for the compiler implementations. Roy Goldfinger and I were working both within IBM and within the Short Range Committee to reduce the differences between Commercial Translator and COBOL, allowing the former to have extra features, particularly the computational forms of FORTRAN. As a result, the GUIDE organization was told, on 1960 January 27, that IBM would include basic COBOL in Commercial Translator. The February 15 survey by the SHARE organization showed Commercial Translator as IBM's version of COBOL (oddly, IBM claimed only 80% machine independence, Honeywell declined to quote on FACT because it was for a single machine, but Wegstein claimed 100% for COBOL).

At the February 17 meeting of SHARE (XIV), Al Harmon, Manager of Applied Programming (and my superior), said: "It appears that time schedules for achieving a version of COBOL that will be satisfactory for all existing and proposed computers would unduly delay IBM's production of processors for Commercial Translator. We are revising our present Commercial Translator manual to represent our best solution to these problems. Our intentions are to revise the Commercial Translator language to include new developments, both from our own efforts and those of the COBOL committee". My verbiage for the official IBM position, announced in *Datamation* magazine, was "The Commercial Translator is being reworked as nearly in the COBOL spirit as possible . . . to ensure that the end result will be a single workable language for data processing". The accent on *workable* came from the Short Range Committee's reluctance to admit the many demonstrated logical flaws that we found in their specifications. Joe Cunningham reported, in 1963 March, that when the total cleanup had been made, the syntax was just as definable as ALGOL, but the semantics were prone to ambiguities.

All of this was very noble and elder-statesmanly. The only problem was that there were *two* versions of Commercial Translator within IBM: the one that Tom Glans, Roy Goldfinger and I specified to merge into COBOL, or perhaps even reconcile to identity; and the other that was written by Barry Gordon, that diverged. Because Gordon was in charge of compiler implementation, our good intention came to naught. Seven pages of differences between the two versions were compiled. I recall trying for free form in the statements in order to avoid punch card limitations, which would be quite handy for terminals today.

At the 6th meeting of CODASYL on 1960 April 7, NCR and General Electric announced their intentions to build COBOL compilers. The latter would make it part of GECOM, which also had algebraic statements like Commercial Translator (antedating PL/I), and a tabular structure facility. Because new manufacturers were asking for participation, Dr. Hopper and I were discharged as advisors, to avoid any appearance of partisanship. In May, Jack Jones of the Air Materiel Command announced the start of an UNIVAC 1105 COBOL. By September, there were 11 manufacturers represented on CODASYL and all had indicated that they would supply compilers (the original six plus Bendix, CDC, GE, NCR and Philco). C.G. Holland-Martin of International Computers and Tabulators had sent a representative to the first CODASYL meeting, and ICT (now ICL) announced a COBOL compiler in October, superseding their own CODEL language.

The internal dissent at IBM kept building, and outside pressures were felt from the user groups. The problem escalated to T.V. Learson, now IBM's Chairman of the Board, who solved it in the style of one who has access to sufficient spendable money. IBM would do *both*, and work toward reconciliation. Accordingly, Al Harmon told the 1960 May 17 meeting of GUIDE that IBM would supply COBOL compilers for the 705 II (without IOCS), the 705 III, 7080, 7070, and 709/90, but declined to give delivery dates. He outlined the two-phase solution of Learson: first a modified Commercial Translator and then a conversion to COBOL. This was most unsatisfactory to GUIDE, which resolved that IBM should stick to its original statement, and that there should be only one compiler per machine, with COBOL an integral part of Commercial Translator.

IBM did not do so, however. Not because it *would* not, but because it *could* not. Roy Goldfinger made an extensive comparison in 1960 July between COBOL and the version of Commercial Translator implemented by Barry Gordon, showing that the original objective was missed by a wide margin. To give IBM management their due, they were honestly chagrined to find that good intentions do not guarantee compatibility in programming languages!

Danny Goldstein of UNIVAC, one of the six members of the Short Range Task Group, prepares to run at the 1960 December 6 compatibility demonstration.

Difficulties persisted. IBM finally announced their COBOL production schedules on 1960 October 1, but it was not until 1962 September that the success of COBOL was sufficiently apparent for Bob Ruthrauff to tell SHARE XIX that "We intend to make COBOL our development language and plan no further development of the Commercial Translator language itself". IBM did not want to put Commercial Translator under the 7090 operating system, but said it would negotiate with diehard users. (Actually, the situation would never have come to this had not the 7090 compiler for Commercial Translator, done by a West Coast group under the direction of Dr. Richard Talmadge, been so good compared to the COBOL compilers existing then.) In 1963 February, SHARE abandoned all hope for Commercial Translator and prepared to switch existing programs over to COBOL via translation routines and some hand work. Honeywell went through much the same difficulties and extra expense; FACT was completed and customers were supported, despite their parallel support of COBOL.

VICTORY?

The participants in the COBOL effort did not disdain publicity. The *New York Times* of 1960 August 26 announced RCA's victory in the "Computer Translating Race", although the language was not full COBOL by any means, nor was the compiler released until December, when RCA and UNIVAC held a joint compatibility demonstration for the public. Both computers operated the test programs identically, UNIVAC II on December 6 and RCA 501 on December 7. Nevertheless, *Datamation* announced that RCA was "making publicity capital out of The Sacred Project" and had "made off with the first publicity marbles". IBM received similar publicity with the COBOL 61 compiler on the 1410.

QUALITY PROBLEMS

In general, the compiling techniques of early COBOL processors were primitive, resulting in very low compilation rates. Navy evaluations reported in 1962 May showed five compilers ranging from 3 to 11 statements per minute. Measurements in mid-1964 showed a range of 11 to 1000 statements per minute. Hardware did not get that much faster in two years, so we must conclude that we found out how to build compilers better. At this time, it was first noticed how drastically compiling rate varied with the size of store (memory) available, and how much variance there was in compiling cost, which ranged in this study from $0.23 to $18.91 per statement. Presently we are in the range of several thousands of statements per minute, and the costs are very much lower.

STANDARDS

Publications via the CODASYL route were COBOL 60, COBOL 61, COBOL 61 Extended, COBOL 65, CODASYL COBOL Journal of Development-69, and CODASYL COBOL Journal of Development-70. The original intent was to update yearly, but this was changed with the advent of formal standardization in early 1963. The CODASYL COBOL committee concentrated on development, leaving the intermediate and formal standardization to the standards bodies. The European Computer Manufacturers Association (ECMA) set up Technical Committee 6 to deal with COBOL, corresponding to American National Standards Committee X3.4.4, which was responsible for developing a COBOL standard (Howard Bromberg, Chairman). These two committees worked jointly to produce twelve COBOL Information Bulletins.

Under the Brooks Bill, US Public Law 89-306, Federal Information Processing Standards are the three-way responsibility of the National Bureau of Standards (NBS), the Bureau of the Budget (now Office of Management and Budget), and the General Services Administration. The first man in the NBS position was Norman Ream, previously of Lockheed. Short funding at NBS prevented the monitoring and measuring efforts scheduled there originally. These difficulties were surmounted when Ream was appointed Special Assistant to the Secretary of the Navy, for he was imbued with the idea that standards would not be too effective without a means of checking conformity. Many compilers were claimed to be COBOL, but, without a Truth In Packaging law or detection device, the user was sometimes misled. Ream's ingenious solution was to call back to active service Commander Grace Hopper, USNR (Ret) to lead the accelerated effort in COBOL standardization for the Navy. Grace was to return to active duty for six months starting 1967 August 1. Only one thing was wrong with the memo announcing this move: she did not get out in 1968, nor in 1969, nor in 1970, when her orders were reissued to read "indefinite". As a result, the Navy has constructed comprehensive COBOL certifiers that are available to anyone.

By being brought under standardization control, COBOL could profit by being related to other information processing standards. To give an example, CODASYL had an early report on the 1960 January 13 meeting of the American Standards Association, at which Committee X3, on Computers and Information

Chairman Phillips and the compiler builders: Howard Bromberg of RCA on the left, and Dr. Grace Hopper of UNIVAC on the right (at the compatibility demonstration).

Processing, was authorized. Character sets were very prominent in the discussion. FLOW-MATIC had 63 characters available, due to the UNIVAC computer, whereas IBM was limited to 48, due to the punch card set at that time. Furthermore, the collating (ordering) sequences were different between the several equipments, and none of the existing sequences survived in the American Standard Code for Information Interchange (ASCII). Yet COBOL has statements in the language which give different actions depending on the collating sequence of the hardware! For reasons unknown to me, CODASYL chose to ignore this problem, and it is still not resolved.

A COBOL specification has been adopted by the International Standards Organization, which refers to standards as Recommendations. Through the efforts of an international editing committee, and careful arrangements by the standards bodies, it is the same as the American National Standard. ECMA also contributed heavily with a formal description of the syntax of COBOL, which was invaluable in reducing ambiguities and validating constructs.

WHAT NOW?

Here we are, beginning a new decade with a regularized COBOL world in both French and English. One might think that the future would hold no surprises. Theoretically, COBOL could continue to maintain a prominent position in computer usage. There is an organizational structure for development, and several standardization bodies to normalize the changes and additions as they are developed. Practically, however, there are some strong factors opposed to its immortality:

- PL/I, a competing language, has been introduced and has attained substantial usage. PL/I also has development and standardizing bodies, at least in the US and in ECMA. It appears that COBOL programs are translatable to PL/I programs with some difficulty, but not too much more than from FORTRAN II to IV programs.
- Although COBOL now has additions for data communication and data manipulation, they are appended to the same old COBOL structure originated for a uniprogramming environment. As I warned in an address to the 10th Anniversary Meeting of CODASYL, data communication and manipulation are but different aspects of general data movement, and should be unified and common to *all* programming languages that must coexist in the same multiprogramming environment. PL/I, COBOL, and FORTRAN, having different development committees, contain data communication facilities that are not common or similar, and yet they must coexist under a single operating system. As the data base becomes king, there is really no need for more than one data procedure language. The user pays heavily for this language plurality.
- The concept of levels may prove vitiating in a communications world. If interchange is stressed too heavily, the tendency will be to write programs in the lowest level of COBOL, in order to assure the widest usability on a maximum number of computers. This is akin to Gresham's Law, and low-level programs will drive out high-level capability. This would be unfortunate, for those lower levels are really unnecessary. The only (and original) reason for their existence is to permit COBOL compilation on small computers, which is something that small computers are particularly unsuited for. It is far better to use the full power of a high level of COBOL, hook up by communications to a remote compilation service, and get back an object program, the running of which does not tax machine capacity the way compilation does.
- Finally, COBOL and the other languages all possess a serious defect for public usage of data: they carry the data description in the program. For interchange and multiple use of public data, the data must be self-descriptive, and the description must be carried along with the data on the storage medium (i.e., disk or tape). Some corresponding change in the structure of these languages is certain.

However, these technical difficulties should not deter the working user from extensive use of the COBOL language. For every successful user there are still several floundering in the morass of emulation because they just could not bring themselves to break away from machine language. The problems of program transferability are great, and the losses from not using transferable languages such as COBOL run well over $1 billion each year. We find countless examples of programmers who have moved on, leaving programs that are undecipherable for maintenance or modification. Had they been written in COBOL, they would have had more than zero salvage value.

In my opinion, emulation is a crutch that never lets you get over the handicap. Fight wastage of human resources; COBOL is a great weapon.