PROGRAMMING LANGUAGE STANDARDS -- WHO NEEDS THEM?

George N. Baird
Paul Oliver

ADPE Selection Office
Department of the Navy
Washington, D. C. 20376

ABSTRACT

The programming language standards used in the United States are produced and maintained by the American National Standards Institute. The standards produced by ANSI are voluntary in that they can be either (ANSI) adopted or ignored by the community which they effect. The first programming language to be standardized was FORTRAN in 1966, followed by COBOL in 1968. The COBOL standard was updated and revised in 1974. The proposed revision to the FORTRAN standard is also close to being adopted.

We can now take a look at what impact that language standardization has had on the user community. This paper addresses the standardization process as reviewed by the COBOL, FORTRAN, and PL/I committees. COBOL will be addressed in greater detail due to the timing and impact of the revised COBOL standard.

The problem with the standardization process as a whole will also be addressed.

INTRODUCTION

The purpose of this paper is to look at the results of the standardization of programming languages over the past ten years and attempt to determine their impact on the computing community. The American National Standards Institute (ANSI), Incorporated is responsible for the production and maintenance of national standards. These standards are voluntary in that they can be either adopted by members of the using community or ignored. In the case of programming languages, the Federal Government has encouraged the implementation of COBOL through FIPS PUB 21[1] and its successor FIPS PUB 21-1[2], and will probably follow suit with PL/I, BASIC, and the soon to be revised FORTRAN standard.

Programming language standards are produced/maintained by a series of Technical Committees called American National Standard Technical Committee (ANSC). ANSC X3J1 is responsible for PL/I, ANSC X3J3 is responsible for FORTRAN, and ANSC X3J4 is responsible for COBOL.

Computer software standards adopted and published by ANSI must be reviewed by the technical committee responsible for that standard within five years of their acceptance. The committee then must do one of three things: reaffirm the standard as being adequate; withdraw the standard as not being necessary or useful; or revise the standard as it deems appropriate.

The guidelines for revising standards suggest that compatability between previous versions of a standard is highly desirable. However,

when a committee is reviewing a standard for the purpose of updating it these guidelines are not always given the appropriate priority.

FORTRAN was first standardized in 1966[3]. COBOL followed suit by being standardized in 1968[4]. The COBOL standard was revised in 1974[5] and gives us the first opportunity we have had to see how the updating of a programming language standard will impact the user community. PL/I[6] and BASIC[7] are close to being adopted as national standards.

FORTRAN and PL/I will be mentioned lightly as to the method in which they have been developed and subsequently subjected to the standardization process. COBOL will be examined in depth from the pre-standardization days to a look at what the 3rd COBOL standard might look like, so as to determine the problems uncovered to date regarding its standardization.

PROBLEMS WITH STANDARDIZATION

The standardization of COBOL is accomplished by ANSC X3J4. The sources of language specifications for the national standard are the CODASYL[8] COBOL Journal of Development (JOD)[9] and the version of ANSI COBOL being revised. This is by an agreement between ANSC X3 and the CODASYL Executive Committee to use only the specifications published in the JOD as input to the national standard. This sometimes precludes the inclusion of commonly implemented language features in the national COBOL standard merely because they have not yet been recognized by CODASYL and placed in the JOD. Thus, the national standard may exclude many language features which have long been part of a de facto standard

because of their implementation by the various producers of COBOL compilers. As a result, there is a problem of restrictiveness in the standardization of COBOL. A second problem has to do with the evolutionary growth of COBOL. (It has been suggested that revolutionary growth might more accurately describe the current changes being made to COBOL by CODASYL.) There is no real assurance that as new COBOL specifications are developed they will remain compatible with previous specifications. There are two reasons for this: (1) a language element/function was ill defined to begin with and therefore subject to implementor definition; (2) incompatible changes to COBOL are introduced leaving both users and implementors with conversion problems. The net effect of this process will be a major conversion every five or so years when compilers are introduced which support the latest COBOL standard.

The X3J1 group which is in charge of the standardization of PL/I will not be bound by the limitations imposed on the COBOL group with regard to sources of language features to be included in subsequent versions of the standard. Presumably, X3J1 will be free to adopt features from any source it so desires. The second problem is however likely to plague the PL/I standard almost as much as it has COBOL. We say "almost" because PL/I has had a period of exposure which should contribute to its stabilization. Added to this is the controversy over the form of the PL/I standard. Basically, the issue is whether a standard should be written for users of a language or for implementors of compilers. Both positions have merit, but it should be pointed out

that user guides are generally provided by vendors as well as authors, and need not have the precision required for implementation.

Then we have the FORTRAN standard, which has shown a remarkable degree of stability. Unfortunately, this stability has also resulted in a standard which has been left far behind by most implementations. FORTRAN users have eagerly used the many extensions introduced by producers of FORTRAN compilers, and with good reason, since many of these extensions are useful.

Are there any common issues here? We think there are several. The first is the role of the Federal Government in the standards setting process. This role has been active and supportive in the case of COBOL. The result has been an evolving standard which creates problems in the very process of evolution. The Government's attitude toward FORTRAN has been one of benign neglect, and the standard has been stable albeit stagnant. The Government's interest in PL/I has been slight, and its attitute has at times bordered on the antagonistic. The PL/I standard has been long in coming and its acceptance is still questionable. (It should be emphasized that in referring to "the Government" we mean the entire spectrum of users and policy makers; we do <u>not</u> mean any given agency, nor do we mean to imply a concerted position on the part of "the Government".) Are these true causes and effects, or simply coincidences with no casual relationships?

Secondly, the industry as a whole needs to re-examine what is the purpose of standards and, more importantly, what is meant by a "standard". The stated purpose of language standards is to assure

the portability of programs; yet the user community spends millions of dollars annually in converting "portable" COBOL programs. One of the reasons for this is that no one was really implementing the COBOL standard. But this is not the only reason. A language standard is necessary to portability, but not nearly sufficient. What we are doing about standardizing the entire interface between the application programmer and the computer system? Additionally, is it fair to refer to something which changes every few years a "standard"? Perhaps we should re-examine our entire approach at language definition to see if we can define more stable languages, perhaps with their own self-extending capabilities.

COBOL STANDARDIZATION -- THE ROLE OF THE AMERICAN NATIONAL STANDARDS INSTITUTE AND CODASYL

The American National Standards Institute (ANSI) Technical Committee X3J4 is responsible for the definition of the COBOL language standard. There are, as indicated earlier, restrictions as to the source from which new COBOL language elements may be selected. The primary source is the documentation of specifications produced by the Conference on Data Systems Languages (CODASYL).

CODASYL is responsible for the development of the COBOL language through one of its subcommittees, the Programming Languages Committee (PLC). The COBOL specifications are made public through the CODASYL COBOL Journal of Development, which contains the official definition of COBOL.

COBOL was initially standardized in 1968. Prior to that time,

compilers were implemented from specifications published by CODASYL and implementor-defined language elements. There were many problems due to the ways in which the CODASYL language elements were implemented. Implementors violated the semantic and syntax rules at will. There was a reason for this; the CODASYL specifications were changing many times each year. As a result, there was no strong desire to attempt to implement a moving target. This resulted in a group of compilers which were developed from the same source language specifications but which were incompatible.

The 1968 COBOL Standard (X3.23-1968) was derived solely from the CODASYL documents as of December 1967. The use of the 1968 COBOL Standard caused problems for the writers of both COBOL compilers and application porgrams. Older compilers generally did not support COBOL as defined in the standard. Therefore, most compilers had to be modified to accept standard source programs. The same problem existed with source programs produced prior to the availability of compilers implemented in accordance with the standard. As a result, converting to COBOL 1968 was, in some cases, so severe a change that language conversion programs had to be provided by implementors. When X3J4 took on the review of COBOL 68 it was obvious that, due to the shortcomings of the 1968 Standard, it needed to be revised. Drawing from the latest CODASYL specifications, COBOL 74 was produced.

The 1974 COBOL Standard will logically replace its predecessor and was derived solely from the CODASYL specifications of December 1971. This resulted in the standard consisting of more state-of-the-art

features but also caused some major incompatibilities with the 1968 specification. As a result of the standardization process, there are (and will be) differences between compilers.

TOWARD DEVELOPMENT OF A PL/I STANDARD

PL/I (short for "Programming Language No. 1") is the product of a team set up by the FORTRAN users of the SHARE organization. It was first described in 1965 by Radin and Rogoway, although its name at the time was NPL (for "New Programming Language"). The name was changed to PL/I when the National Physics Laboratory of Great Britain objected to the use of its initials.

The original SHARE effort produced a complete specification of the language before any compiler was produced. The very first report was really a union of FORTRAN and COBOL and reflected more an attempt at consolidating the past than anticipating future needs. A second report, a much improved effort, was circulated for comment prior to the third release, which represented the first official definition of the NPL.

The American National Standards Institute and the European Computer Manufacturers' Association (ECMA) have had committees working jointly since 1969 to produce a PL/I Standard, and this effort is now nearing completion. A draft proposed standard for PL/I was published in February 1975 under the aegis of the Computer and Business Equipment Manufacturers Association (CBEMA).

This draft standard has undergone a period of public review during which comments regarding it have been considered. It is anticipated that the draft standard, revised as necessary, will be submitted to the

American National Standards Committee X3 for letter ballot. Upon completion of the ballot, the proposal will be forwarded to the American National Standards Institute for approval as an American National Standard. Concurrently, the draft standard is being considered by the European Computer Manufacturers' Association and the International Organization for Standardization (ISO) for approval as an ECMA and an International Standard, respectively.

Despite the general anticipation of approval, acceptance of this standard by ANSI, ECMA, and ISO is not a foregone conclusion, as there is a substantial nucleus of opposition to it. This opposition stems less from the merits of the language itself, which is essentially the same PL/I familiar to most IBM users, but from the definitional method of Standard PL/I. This method has attempted to close the gap that has always existed in the industry between language definition theory and practice. In the process the ANSI committee responsible for PL/I (X3J1) has striven to combine the precision and completeness required by a formal definition with the lucidity and naturalness essential to general acceptance. Most observers agree the committee has done an admirable job with respect to precision and completeness. Many feel it has completely failed with respect to lucidity and naturalness.

The method of definition is such that semantics are defined by showing how language interpretation would affect the workings of an abstract machine. The abstract machine is so defined that, when offered the representation of a purported PL/I program, it will determine whether it is valid, and if so will execute the program to cause

certain changes in the state of the machine. The concrete syntax of Standard PL/I is defined by production rules expressed in a metalanguage not dissimiliar from that used to define ALGOL 60. The abstract syntax of PL/I is the syntax of programs in a representation that is convenient for the definition of semantics. A translator is defined that converts program segments from concrete syntax representation to abstract syntax representation. Finally, an abstract machine, the interpreter, is used to define the semantics of a given program statement by executing its abstract syntax representation according to a prescribed set of rules. Errors in a given program segment may thus be found at three distinct levels:

(1) Concrete syntax rules render the statement

```
GET LIST (A(I,J) DO I = 1 TO M,N);
```

illegal, since an extra pair of parentheses is required to determine whether M is an input variable or part of the DO-loop.

(2) The statement

```
DECLARE X FLOAT FIXED;
```

is correct according to the concrete syntax, but fails to satisfy the abstract syntax during the translation phrase (it gives X the conflicting attributes of both fixed and floating point).

(3) The question of whether all 5 elements of A are set to 0, where A is allocated, by the statement

```
DECLARE A (5) INITIAL (0)
```

cannot be resolved until the interpretation phrase, as it is a semantic issued.

A complete description of the definitional method used for the PL/I standard and a complete discussion of the ensuing controversy are beyond the scope of this paper. A flavor of the nature of the critics' objections may be obtained by the realization that the answer to the question posed by the third example required the performance of interpretation steps that fill nearly two full pages of the standard document (small print at that). Surely, say the critics, this is too much to find out that only the first element, A(1), is initialized.

Despite the controversy, it is safe to say that some form of a PL/I standard will soon be available. It is also safe to say that despite, the controversy it has evoked, the draft proposed Standard PL/I is a landmark document that will have a significant impact on the industry.

FORTRAN STANDARDIZATION

The standardization process for FORTRAN is unlike that of both COBOL and PL/I. There is no 'other' committee which produces the language specifications for FORTRAN, as CODASYL does for COBOL. There are no strong ties or dependencies between X3J3 and the international community as is for PL/I. (ANSC X3J3 is responsible for the maintenance of the FORTRAN standard.)

X3J3 has been working most diligently toward a revised FORTRAN Standard since 1966 when the first FORTRAN Standard was approved. The

language elements which will makeup the revised FORTRAN Standard are drawn primarily from the user community and particularly from specific implementations. This in effect takes de facto standards (supported by several implementations) and includes them in the national standard.

An important consideration in the preparation of the revision to the FORTRAN Standard was the minimizing of conflicts or incompatibilities with the previous standard. The new standard includes changes which created incompatibilities only when the change corrected a previous error or added significantly to the FORTRAN language. An additional consideration concerning the approval of incompatible changes was based on the number of programs it might impact. There are only 10 known conflicts between the 1966 FORTRAN and its proposed revision, one of which involves the addition of several functions which may cause some user-defined words to be modified. (Contrast that with 56 incompatible differences between COBOL 68 and COBOL 74.)

Current FORTRAN compilers which will accept a FORTRAN program based on the 1966 FORTRAN Standard will merely have to be extended to accept programs containing new features from the revised FORTRAN standard.

THE ROLE OF THE FEDERAL GOVERNMENT

The role of the Federal Government in the development of the COBOL language has been extensive and decisive. COBOL is the "standard" programming language in the Government. Furthermore, COBOL compilers represent the first category of software which is measured for conformance to a standard. The role of the Government in the validation of

COBOL compilers is particularly significant.

The concept of validating COBOL was established 15 January 1963. This was the first meeting of the American Standards Association (ASA) Committee* Task Group for Processor Documentation and COBOL. The program of work for the Task Group included the writing of test problems which could be used to determine that COBOL compilers did indeed support the features of the COBOL language. A working group was established for creating the test problems. In April 1967, the Air Force issued a contract for a system to be designed and implemented which could be used in testing a compiler against the standard. In August of 1967, the Special Assistant to the Secretary of the Navy created a task group to encourage the use of COBOL throughout the Navy. Being aware of both the ASA Task Group and Air Force efforts, a project was established to determine the feasibility of validating COBOL compilers. After examinging the information and test programs available at that time, the first working set of routines was produced by the Navy Programming Languages Group under the direction of Captain Grace Hopper, USNR. Because of the favorable comments received on this initial work, the Navy decided to continue the effort. After steady development and testing for a year, Version 4 of the Navy COBOL Audit Routines was released in December 1969. The routines consisted of 55 programs (18,000 card images) capable of testing the full standard.

In December 1970, the Deputy Comptroller for Data Automation

*The American Standards Association (ASA) has since changed its name to the American National Standards Institute, Incorporated (ANSI). This committee is known as X3J4 COBOL Standardization and Maintenance.

in the Office of the Secretary of Defense asked the Navy to create the Department of Defense Compiler Validation System for COBOL, taking advantage of (1) the better features of both the Navy COBOL Audit Routines (Version 4) and the Air Force CCVS and, (2) the four years of experience in designing and implementing audit routines on various systems as well as the actual validation of compilers for procurement purposes. Responsibility for ensuring the appropriate validation of COBOL compilers throughout the Department of Defense was given to the Department of the Navy's Information Systems Division (OP-91), and the Central COBOL Compiler Testing Facility (CCCTF) was created to discharge this responsibility.

Subsequently, the National Bureau of Standards (NBS), which has the responsibility for the development and maintenance of Federal ADP Standards, delegated to the Department of Defense and thereby to the Department of the Navy the responsibility for the operation of a Government-wide COBOL Compiler Testing Service, which has replaced the CCCTF. This responsibility is discharged by the Federal COBOL Compiler Testing Service (FCCTS), an activity of the Software Development Division of the Department of the Navy Automatic Data Processing Equipment Selection Office (ADPESO) through the implementation and maintenance of the COBOL Compiler Validation System (CCVS)[10]. The first enforcement effort on the validation of COBOL compilers was the issuance by GSA of a Federal Property Management Regulation (FPMR)[11]. This regulation requires all compilers brought into the Federal inventory to be validated, and also that any derivations must be corrected within 12 calendar months.

COBOL - FROM PRE-STANDARDIZATION DAYS UNTIL THE PRESENT

The problem deals with the conversion of programs between different versions of the American National Standard COBOL (i.e., between X3.23-1968 and X3.23-1974). The cost of programmer time is becoming more expensive while software and computers in general are either decreasing or leveling off in price. This heavily impacts the use of COBOL when conversion is considered. For example, if a program based on COBOL 68 merely had to be re-compiled using a compiler based on COBOL 74 then a certain level of personnel time would be involved. This could be easily accomplished by a trainee or a programming librarian.

If any conversion is involved, the amount of personnel time will increase drastically due to the human interaction of making changes to the source program during several compilations. The process may be shortened if a form of a source program translator is available. Still an individual would have to run the source program through the translator, check the output of the translator and finally run the corrected source program through the compiler. (There may be syntax changes which are missed by the translator and would require human intervention.) For semantic differences, further debugging may be required even if the areas to be changed are easily located.

The ideal situation, (and the most cost effective) at this point would be to simply recompile without any conversion. The 54 page COBOL conversion document produced by the National Bureau of Standards (FIPS PUB 43)[12] shows that this is not the case between COBOL 68 and COBOL 74. Based on the premise that standards exist to protect our

programming investment and at the same time to save the user money, the following is presented to show the precise problem with the current COBOL development/standardization effort.

COBOL is the most widely used high level programming language in this country. As a result there are many installations which are dependent on both the development and standardization of COBOL. The community as a whole has welcomed the standardization of COBOL through the acceptance of the 1968 COBOL Standard. The introduction of the 1974 COBOL Standard has left users decidely frustrated. The reason for this is the degree of incompatibility between the 1968 and 1974 COBOL Standards. Further, the implication of what the next COBOL Standard might look like suggests that unnecessary incompatibilities will exist between the current 1974 COBOL Standard and its successor.

The comments on incompatibility in the above paragraph seems to defy the purpose of having standards as well as the purpose which CODASYL originally intended COBOL to accomplish:

> "In developing data processing systems for existing computers, it is important that these systems be capable of processing on future, more powerful computers of any manufacturer, with a minimum of conversion costs."

The above quote is from the objectives of the CODASYL COBOL Journal of Development, page 1-1-1, paragraph 1.1. This objective can only be met if the development of COBOL is controlled so that there is a compatible evolution of the language as it is enhanced and encompasses more capabilities and functions.

The Government has been a heavy user of COBOL for reasons of

compatibility between systems. This has been primarily due to (1) procurement practices (which can cause the upgrading of a system to result in its replacement by a totally different computer); (2) sharing of software; and (3) backup purposes. Compatibility between COBOL compilers is a must. This has been partially achieved through the formal standardization of COBOL, the use of Standard COBOL as a programming language and the validation of compilers to determine whether they conform to the specifications contained in the COBOL Standard. As a result of both the national standards program (ANSI) and the standards programs within the Government, a user can, to a degree, achieve the aforementioned objectives set forth by CODASYL. The problem addressed here began as the COBOL community attempted to implement and use COBOL as defined in the 1974 COBOL Standard. The basic problem has to do with the incompatible development of COBOL since 1967.

In justifying what has been stated in the previous three paragraphs, the impact of using COBOL must be described from pre-standardization days through the next COBOL Standard (the update to COBOL 1974). Discussion should be prefaced with the ground rules used to date in regard to both COBOL and its sponsor CODASYL. As was stated earlier, ANSC X3J4 through its direction from X3, CBEMA, etc. recognize the CODASYL COBOL Journal of Development (JOD) produced by the Programming Language Committee (PLC) as the sole source for COBOL language elements which were candidates for inclusion in the 1968 Standard. For the 1974 Standard language elements from both the JOD and the 1968 Standard were eligible for inclusion.

Pre-1968 Standard

COBOL compilers were implemented from the CODASYL specifications as well as implementor defined constructs. The problem at this point was due to the ways in which the CODASYL language elements were implemented. Portions of statements were implemented; semantic and syntax rules were violated. This resulted in compilers developed from the same language specification but which were basically incompatible.

1968 Standard (X3.23-1968)

The 1968 COBOL Standard was derived solely from CODASYL COBOL JOD updated through 1968. The use of COBOL 1968 caused problems for the writers of both COBOL compilers and COBOL programs. Compilers previously written generally did not support the modules as defined in the standard due to the implementation techniques and language element selection techniques described above. Therefore, most compilers had to be modified to accept source programs based on the 1968 COBOL Standard due to the method by which language elements from the JOD had been implemented in a modified form to suit the implementor. As a result, converting to COBOL 1968 was so severe that in some cases language conversion programs had to be provided by implementors to take the user from the existing compiler to the new COBOL 1968 compiler.

The justification for the "cost" of conversion was that future conversions would be unnecessary whether it was to a new compiler for the current system or to an entirely foreign system. Also, if manufacturers preferred supportong only one compiler, the impetus was to support a compiler based on the 1968 COBOL Standard due to Government

procurement policy (FIPS PUB 21) and the ability to demonstrate compatibility with other systems.

1974 Standard (X3.23-1974)

The 1974 COBOL Standard was derived solely from the CODASYL COBOL JOD updated through 1971. Please note that the ground rules stated above indicated that the 1974 COBOL Standard would be created from elements in both the 1968 COBOL Standard and the CODASYL COBOL JOD. The reason for this is simply to allow a language element dropped by CODASYL to be retained in a future standard by ANSI for compatibility purposes. During development of the 1974 COBOL Standard this was not done and as a result several language elements disappeared from the COBOL Standard during its revision.

There are some disturbing differences between the 1968 and 1974 COBOL Standards which are relevant to the topic of this dicussion. Some changes which caused incompatibilities were necessary due to lack of an adequate definition of language elements. Other changes which cause incompatibilities were due to bad planning and/or lack of coordination on the part of the COBOL development group. Below are a few of the incompatible differences between COBOL 1968 and COBOL 1974 and a relevant comment for each:

(a) Random Access Module - The key by which records were stored and retrieved was implementor defined. This precluded any hope of compatibility between implementations. This module was replaced in the 1974 COBOL Standard by Relative I-O and Indexed I-O modules which are very explicit about the key which is used to store

and retrieve records. There should be little or no compatibility problems between implementations supporting COBOL 1974. However, conversion may be severe for programs based on the 1968 specification.

(b) EXAMINE Statement - Deleted and replaced by the INSPECT statement. This was one of the most illogical changes made. The EXAMINE statement is used quite often and will require conversion. The INSPECT statement gives the user additional capability that the EXAMINE statement did not permit. However, the EXAMINE statement in the 1968 COBOL Standard could have been modified, in a compatible way, to accomplish everything the INSPECT statement is capable of, and thereby precluded any problems with conversion.

(c) NOTE Statement - Deleted in favor of a technique which was more versatile to use and cheaper to implement compared to the NOTE statement. This requires conversion of any program in which the programmer "wisely" documented his program with comments in the Procedure Division. It could certainly be debated whether the cost of conversion will outweigh the advantages of having a more versatile comment capability. It could also be debated as to whether in this case the development group dropped the ball by deleting the NOTE statement or X3J4 fumbled it when they did not retain the NOTE statement as well as include the new comment capability.

(d) REMARKS paragraph in the IDENTIFICATION DIVISION - Deleted in favor of the new technique in (c) above. This deletion begs for justification. There are several paragraphs in the Identification Division which contain comment entries as did the REMARKS paragraph. The conversion in this case is simply the removal of the word

"REMARKS" from the source program. The comment entry previously associated with the REMARKS paragraph would then belong to the previous paragraph which would not contain additional comments. This conversion problem borders on the ridiculous.

There were many other incompatible changes between the 1968 and 1974 COBOL standards, but the four above have demonstrated both unjustified changes (b, c, and d) as well as poor planning and ill definition (certainly a and possibly b and c). Appendix B of X3.23-1974 American National Standard Programming Language COBOL contains a list of changes made to COBOL 68 to produce COBOL 74. 44 changes render existing statements incompatible with COBOL 74 and 12 elements previously implementor defined are now defined in COBOL 74.

There will be a conversion cost associated with using the 1974 COBOL Standard. Part of it is anticipated and expected in areas whose definition was left to the implementor of the compiler as was the old Random Access module. It would have been better if these areas had either been completely defined in the beginning or not included in the standard. The other part of the conversion costs can only be explained as being associated with change for the sake of change which could have been avoided with a bit of discretion on the part of the CODASYL and/or X3J4.

The justification of converting from the 1968 COBOL Standard to the 1974 COBOL Standard could be stated in terms of better defined language elements as in converting from the old Random Access to either Relative I-O or Indexed I-O. But again there will be additional costs due to the unnecessary conversion that is required.

The Next COBOL Standard

There are several questions regarding the next COBOL Standard. It should certainly be a superset of the 1974 COBOL Standard. Source programs acceptable to a 1974 COBOL Compiler should be acceptable to a compiler based on the next standard with a minimum of changes required. The only exception should be changes with defined language elements/ actions previously left either to the discretion of the implementor or undefined. If this cannot be accomplished then a question arises as to whether the 1974 COBOL Standard (FIPS PUB 21-1) should be updated.

The evolution and compatibility problems are shown by recent actions taken by CODASYL in regard to changes to the JOD. These changes provide no additional capability to the user, but from a syntax point of view will be unmitigated disaster to the user. The points in question are covered in the PLC minutes from meeting of November 7, 1975. In summary there are five changes which are of concern.

(a) The ALTER statement was deleted. Regardless of whether the ALTER statement is good, bad, or indifferent, it has been eliminated and programs using it will have to be converted at some point.

(b) The realignment of the clauses between the ENVIRONMENT and DATA Divisions means that clauses which are currently in the Environment Division will have to be moved to the Data Division and vice versa. This could invalidate 90% of the existing COBOL programs.

(c) The deletion of the 77 level item in the Working Storage section as being redundant with an elementary 01 level item will probably invalidate 95% of existing COBOL programs.

(d) The deletion of numeric procedure names begs for justification. It has been permitted since the first CODASYL report on COBOL was published. There is no way a user could have protected himself from this kind of a change.

(e) The OPEN REVERSED facility was deleted. This permitted a file which is at its end to be read in reversed order (i.e., last record first, next to the last record second, ... first record last).

The above changes collectively will probably impact every COBOL program every written. The quote from the objectives in the JOD mentioned earlier has just become a grossly misleading statement.

Recommendation For COBOL

It appears that as the COBOL language is developed no thought is given to existing compilers, libraries of source programs and the cost of eventually having to modify both compilers and source programs. This is not a criticism of development work which can extend COBOL to accomplish new things, but it is a criticism of the apparent disregard to the amount of time and money some of these meaningless changes are going to cost COBOL users. In this day and age of shrinking budgets the economic impact of language changes should be taken into consideration and justified prior to permitting the changes to be made.

The problem has been adequately defined and unfortunately the solution is not as simple as describing the problem. The next COBOL standard should be a result of the compatible growth and maturing of COBOL 74. The problem associated with controlling this would logically require the cooperation of the two groups involved.

A proposed solution to the problem for COBOL follows and is by no means intended to solve all of the problems associated with the development and standardization of a high level language - but merely a means by which the cost of developing, maintaining and converting compiler/programs and more importantly using COBOL can be reduced.

(a) CODASYL should not make changes (semantic or syntax) that would be incompatible with the current COBOL Standard. The only exception to this would be a language element which contained an implementor defined aspect leading to unpredicatable results. (This would protect our current programming investment and ensure that programs running today would not have to be needlessly converted for use on future compilers/systems.)

(b) X3J4 should not select language elements from the CODASYL JOD which contain an implementor defined aspect which would lead to unpredictable results. (This does not include entries in the Environment Division which require the use of an implementor provided name.)

(c) X3J4 should produce a standard which is compatible with the current COBOL 74 Standard. Areas which are currently implementor defined or lead to unpredictable results should be either defined or deleted.

CONCLUSION

The standardization of programming languages is a workable concept and should be exploited. There are some concrete ground rules which should be imposed in the standardization arena which will preclude the problems described in this paper from being perpetuated in the future.

1. There should be upward compatibility between a programming language standard and its successor. Any deviation from this should be economically justified (X3J3 at least attempted this in revising the FORTRAN standard).
2. Standards should be updated sooner - FORTRAN being the case in point here.
3. More implementation experience and use should be required for language elements being included in a language standard. (COBOL is the worst offender in this case.)

REFERENCE

1. Federal Information Processing Standards Publication 21 COBOL (68) March 15, 1972, U. S. Government Printing Office, Public Documents Department, Washington, D. C. 20402.

2. Federal Information Processing Standards Publication 21-1 COBOL (74) December 1, 1975, U. S. Government Printing Office, Public Documents Department, Washington, D. C. 20402.

3. X3.9-1966 American Standard FORTRAN, American National Standards Institute, Inc., 10 East 40th Street, New York, New York 10016.

4. X3.23-1968 American National Standard COBOL, American National Standards Institute, Inc., 10 East 40th Street, New York, New York 10016.

5. X3.23-1974 American National Programming Language COBOL, American National Standards Institute, Inc., 10 East 40th Street, New York, New York 10016.

6. BSR X3.53 Basic/1-12 1975 February Draft Proposed Standard Programming Language PL/I CBEMA, 1828 L Street, NW, Washington, D. C. 20036.

7. BASIC Draft Proposed Standard Programming Language BASIC, CBEMA, 1828 L Street, NW, Washington, D. C. 20036.

8. Conference on Data System Languages - Non-Profit organization that is responsible for controlling the growth of COBOL.

9. CODASYL COBOL Journal of Development 1972, 110-GP-1C, Technical Services Branch, Department of Supply and Services, 5th Floor, 88 Metcalfe Street, Ottawa, Canada.

10. Baird, G. N. and Cook, M. M., "Experiences in COBOL Compiler Validation", Proc. 1974 NCC, AFIPS Press, Vol. 43, pp. 417-421.

11. Federal Property Management Regulation (FPMR) 101-32.1305-1a COBOL Compiler Validation Federal Register, Vol. 40, No. 221, Friday, November 14, 1975.

12. Federal Information Processing Standards Publication, 12 Aids for COBOL Conversion FIPS PUB 21 to FIPS PUB 21-1, December 1, 1975, U. S. Government Printing Office, Public Documents Department, Washington, D. C. 20402.

BIBLIOGRAPHIC DATA SHEET	1. Report No. FCCTS/TR-77/17	3. Recipient's Accession No.
4. Title and Subtitle: Programming Language Standards Who Needs Them?		5. Report Date: 9 May 1977
		6. (12) 26p.
7. Author(s): George N. Baird and Paul Oliver		8. Performing Organization Rept. No.
9. Performing Organization Name and Address: Software Development Division, ADPE Selection Office, Department of the Navy, Washington, D. C. 20376		10. Project/Task/Work Unit No.
		11. Contract/Grant No.
12. Sponsoring Organization Name and Address: ADPE Selection Office, Department of the Navy, Washington, D. C. 20376		13. Type of Report & Period Covered
		14.

15. Supplementary Notes

16. Abstracts

The programming language standards used in the United States are produced and maintained by the American National Standards Institute. The standards produced by ANSI are voluntary in that they can be either (ANSI) adopted or ignored by the community which they effect. The first programming language to be standardized was FORTRAN in 1966, followed by COBOL in 1968. The COBOL standard was updated and revised in 1974. The proposed revision to the FORTRAN standard is also close to being adopted.

We can now take a look at what the impact that language standardization has had on the user community. This paper addresses the standardization process as reviewed by COBOL, FORTRAN, and PL/I committees. COBOL will be addressed in greater detail due to the timing and impact of the revised COBOL standard.

17. Key Words and Document Analysis. 17a. Descriptors

17b. Identifiers/Open-Ended Terms

408438

17c. COSATI Field/Group 09/02

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release unlimited.	UNCLASSIFIED	26
	20. Security Class (This Page) UNCLASSIFIED	22. Price