REPORT TO SPARC (the Standards Planning and Requirements Committee of ANSI) FROM THE AD HOC COMMITTEE ON OPERATING SYSTEM CONTROL LANGUAGES

Chairman, OSCL, Millard H. Perstein, System Development Corporation, 2500 Colorado Avenue, Santa Monica, California 90406. 1971 July 6.

1. INTRODUCTION

1.1 HISTORY. At the USASI X3.4.2 (Committee on Programming Languages) meeting of 1967 June 27-28, M. H. Perstein presented a proposal that "X3.4.2 shall initiate action to provide an industry-standard input language for top-level control of the computing process, ...". The proposal was presented following a suggestion developed by an internal technical committee at System Development Corporation, Mr. Perstein's employer.

The proposal did not come to a vote at that meeting, but the chairman of X3.4.2, P. Z. Ingerman, appointed Perstein a committee of one to investigate community interest in the matter and to report back to X3.4.2. In addition to direct mailings to persons who might be interested, Perstein sent a news release to the industry and professional press. Because of the inept paraphrasing of the release by one news medium, the propriety of the wide distribution and the use of a news release were questioned at the next meeting of X3.4.2, despite the specific purpose of determining community interest imposed on the Perstein committee of one.

At the next meeting of X3.4.2, 1967 August 22-23, Perstein reported that four favorable responses had been received. At the meeting on 1967 November 1-2, he reported that there was some interest in evidence, but not enough to form a committee. The question remained open.

At the meeting on 1968 January 22-25, Perstein reported a continuing display of interest and requested approval of a news release incorporating a strong letter of support from N. J. Ream, Special Assistant to the Secretary of the Navy. The release was approved by C. A. Phillips, Chairman of X3. Following this meeting, Perstein issued the news release and received many assurances of support in response.

At its meeting of 1968 June 18-19, following some controversy on procedural and jurisdictional matters, X3.4.2 voted to form an ad hoc committee on standard operating system control language (X3.4.2F).

The scope and program of work for X3.4.2F were prepared at the 1968 August 14-15 meeting of X3.4.2 and recommended for approval by X3.4. Perstein noted that he had four volunteers for X3.4.2F, when, as, and if it be formed.

X3.4 approved the scope and program of work of X3.4.2F with a slight change. At the X3.4.2 meeting of 1968 October 21-23, Perstein reported ten volunteers, in addition to himself, were willing to work on X3.4.2F. Ingerman appointed Perstein Chairman of X3.4.2F.

The first meeting of X3.4.2F was held 1969 February 4-5 with 35 people in attendance. Much of this meeting was in the nature of a seminar on various aspects of the need for standards in this area. Work began on the delineation of the functions and the users of an operating system control language. This was reported to X3.4.2 at its meeting 1969 March 5-6.

Since then, USASI has become ANSI, X3 has been reorganized, with X3.4.2F reporting to SPARC, and its name changed to OSCL. Although the reorganization did not dissolve X3.4.2, it has not met since 1969 March 6.

1.2 SEMANTICS.

1.2.1 Operating System Control Language (OSCL). This is the language for controlling the operating system. There must exist, in order for the term to be meaningful, an operating system to be controlled by OSCL or some sort of artificial language. Such systems exhibit a very broad range over the spectrum of sophistication. Somewhere in the middle part of the range there must lie a set of OSCL functions of broad general interest which is narrow enough to be meaningful but broad enough to be useful. If the user approaches a bare machine, one endowed with electricity but devoid of software, he must control the operation of the system by manipulating console switches and loading and unloading peripheral equipment in the proper sequence as well as providing a program to control all aspects of the desired computation. This is probably the low end of the sophistication range. At the other end is the elaborate hardware and software system which approaches the capacity of humans. This is a system in which the manipulation of the system may be very loosely defined and in fact may be arrived at by a conversation between the user and the system to agree on the level of control. Here at the high end, the control language may devolve to the vernacular and may approach natural language in freedom and complexity.

In another sense the spectrum of operating systems control languages covers a range of systems which may be categorized as small to large in size. This range of systems is orthogonal to the sophistication range and the combination of the two ranges leads to a set of systems which vary in at least two dimensions. It is clear that the OSCL requirements of the very small sized systems are probably quite different from the very large sized systems.

It is not clear just where, over these ranges, are the boundaries, within which the proper sphere of control language is to be drawn. Nevertheless, OSCL is the language for controlling a computation system which includes, for the purpose of such control, software characterized by some point in the vast range of sophistication and designed to interpret and respond to such language.

Several aspects of such a language may be candidates for standardization. Such standardization cannot be contemplated until the semantics of control of computation systems can be categorized and defined. It may be that further study will be able to decide what parts of the responses to a control language may also be standardized.

In order to approach this area intelligently certain hypotheses may be posed. These hypotheses, described below in 1.2.2, 1.2.3, and 1.2.4, form the basis which the committee feels must exist if a standard is to be developed. Unless these hypotheses can be clearly validated, standardization cannot exist.

1.2.2 Elementary Functions. There is a feeling in the committee that there exists a set of units of work or of response which represent the repertoire of the computation system. It is not proposed that these elementary functions be quantized or indivisible, but rather these are hypothesized for convenience only.

It is assumed that there is a three step process involved in breaking down the universe of control function into elementary functions. The test of this hypothesis is that the process can actually take place. The steps envisioned are:

1. The universe of function can be broken down into specific functions done in response to a single OSCL command.

2. That there are some responses which occur for more than one command. All the set intersections are considered as candidates for a single elementary function each. All the set differences are equally candidates.

3. For purely esthetic reasons these sets may be further subdivided. In this case the separability of the sets will have to be maintained.

Each of the sets arrived at at the end of step 3 is defined to be an elementary function. It is a further requirement that the set of elementary functions shall be bounded and in fact the total number should be relatively small.

That this can be done for an existing system seems obvious on the surface. That it can be done for all existing systems (or even a fair number of these) is not at all clear. The real test is whether this operation can be extrapolated from existing systems to satisfy the needs of a wide range of users of computation systems.

1.2.3 Functional Group. In the context of operating systems it is possible to hypothesize the existence of a related group of elementary functions to be invoked when the user issues a single command. The proof of this hypothesis must rest on the ability to be able to transform non-command type control functions such as keys, buttons and other physical action devices into the same context as more usual commands.

It is further assumed that it is possible to structure the semantics of commands into the familiar action symbol (sometimes called a verb) and a set of contextual modifiers (usually called operands).

1.2.4 Structure of Commands. This topic refers to the syntax of commands. The hypothesis to be tested here is that when the set of functional groups is known semantically, there can be developed a syntax to incorporate all of them. This includes all of the usual periphery of such syntaxes such as command words, parameters, internal delimiters, terminators and character sets.

If the hypothesis 1.2.3 is proved correct, then this hypothesis can be tested by producing at least one potential structure which satisfies the hypothesis.

1.3 CONCLUSIONS. We have concluded that there is a need for a standard. We have also concluded that there is a clear need for further work on OSCL. We propose that a formal study committee be formed.

It is the strong feeling of the current committee that a standard is much needed and that work should proceed with deliberate speed to do the necessary groundwork. After the study work has been completed, it is expected that there will be a recommendation for one or more standards in this area.

There are a large number of potential OSCL candidates. In fact the current committee did not have sufficient resources to complete exhaustive surveys of OSCLs. None of the OSCLs surveyed so far is considered sufficient as a basis for a standard, and it may be that a more complete survey will not alter this conclusion.

1.4 RECOMMENDED SCOPE OF THE STUDY COMMITTEE. The committee should be charged with codifying and organizing the existing information about operating systems control language with the goal of recommending specific standardization activity. Based on the work of the last two years, the study committee must extend into those areas not yet considered. In particular, the requirements and peculiarities of networks of operating systems must be fully exposed. The interactions of Data Descriptive Languages and Data Base Control Languages with OSCLs must be examined. The relationships of privileged users to the integrity and reliability of the system may well have implications with respect to the OSCL and should be studied. The interactions of hardware and OSCLs must be considered.

1.5 RECOMMENDED PROGRAM OF WORK OF THE STUDY COMMITTEE. Consider the elementary functions that operating systems might perform. Consider those functions illuminated by existing surveys and as many others as are deemed necessary to truly represent all functions that a system might perform for users. Since many alternative sets of elementary functions will be considered, it will be necessary to evaluate different sets to find the most appropriate set.

Is it possible to subset the collection of elementary functions and treat these subsets as whole sub-languages of OSCL? Is there some set of nucleus functions which are mandatory in any OSCL?

Define a way to describe these elementary functions with a minimum of ambiguity and inconsistency. In particular, is there a way of describing these functions such that various systems can successfully offer the same functions?

In particular, examine the relationship of the elementary functions of a Data Base Control Language to the OSCL elementary functions. Also examine the report of the ANSI DDL Study Committee as it relates to OSCL.

Complete the study of the implications of hardware characteristics on control languages. It is apparent that every system in existence has external differences. Are these differences purely cosmetic or does the fundamental underlying system have a real impact on the form of OSCL associated with a system?

Define a method of measuring conformance with the eventual OSCL standard. Such a measure is probably required in the future so that there can be no question as to the requirements that conforming systems must meet.

Define and describe the impact that the external media (typewriters, scopes, etc.) may have on the forms of an OSCL. Consider the necessity or desirability of device dependent language forms.

Is it possible to divorce certain system responses from the OSCL or should all responses be categorized and regularized? Definition in this area is much needed.

In light of the above studies can the following hypotheses be proved or disproved?

1. There exists a set of elementary functions.

2. There exists a reasonable set of user-oriented groupings of these functions.

3. There exists a reasonable user-oriented syntax for commands to invoke these groups.

4. There exists a reasonable set of commands following this syntax.

When all of this study has been accomplished with sufficient thoroughness, is it possible to recommend that one or more standards in the OSCL domain should be promulgated and specifically in which areas?

Determine the impact on users of having to convert to the eventual standard OSCL. This should be estimated for various size systems from a very small to a very large and for a wide variety of applications areas.

2. LANGUAGE/FUNCTION SURVEY

2.1 WHAT WAS SURVEYED (HISTORICAL). In an attempt to satisfy point 1 of the program of work and to determine if an existing control language might be suitable for a standard, the committee developed surveys of the following operating systems and their control languages:

1. GECOS III
2. EXEC VIII
3. Honeywell Mod 4 OS
4. MULTICS
5. PDP - 10 Monitor
6. 360 OS (TSO)
7. 360/DOS
8. 360/TSS
9. Sigma 5/7 BTM

These surveys may be found in their entirety in Section 6.4.

The control langauges of these particular operating systems were chosen as candidates for the survey because it was felt they presented a reasonable sampling of existing systems and were familiar to the surveyors. In addition the majority support both the batch and interactive modes of control in both uniprogramming and multiprogramming environments.

2.2 HOW SURVEYED. An attempt was made to standardize the method of surveying each operating system by having the surveys follow a common outline. General information was provided for the following sections:

1. User/System Interface
2. The Internal Components of the System
3. Initial State of the System
4. Syntax of the Control Language
5. Normal Behavior of the System
6. Interruption of Normal Behavior
7. Privileged Use
8. System Functional Diagram (Beech Tree)

The section entitled "Normal Behavior of the System" contains the lexical structure of the command language. Subsections are provided for:

1. Process Management
2. Program Management
3. File Management
4. Control Language Modification
5. Enquiries

Each subsection contains the pertinent individual commands. Although in most cases no attempt was made to specifically identify associated parametric data, the general functions performed by each command were specified as well as their modes of operation (interactive, non-interactive or both).

In an attempt to provide accuracy and comprehensiveness, each survey was, where practical, prepared by an individual who was familiar with the system being surveyed either through vendor or user association. In general the surveys are a summary of information contained in manuals or other customer documentation.

Although the surveys are reasonably comprehensive in scope and content, each is an independent document. It was immediately obvious to the committee that some means of crossreferencing the data contained in them was necessary. After several methods had been investigated a systems and functions matrix was devised which correlated a composite list of functions against the functions provided by eight of the operating systems surveyed. The resultant matrix revealed some rather interesting information. This will be covered in Sections 2.4 and 2.5.

2.3 WHAT WAS NOT SURVEYED AND WHY. There were two limiting factors governing the breadth of the systems surveyed. First it was necessary that there be someone who had the time and energy to make the survey. Second the committee felt that the larger systems would provide the most information about OSCL.

As a result only existing systems that were well documented, reasonably large, and known to the surveyors were chosen. Of course the surveys were not exhaustive in this area as a number of manufacturers' OSCLs were not included. Network systems, sensor based systems and elaborate multiprocessing systems were also omitted.

2.4 COMMONALITY OF FUNCTIONS. Prior to applying the data contained in the surveys to the systems and functions matrix, a complete function list was produced. This function list was subject to several iterations during the earlier committee meetings and the list used for the matrix was somewhat abbreviated from earlier versions.

After all of the surveys were applied to the matrix the following points were noted:

1. There was a good deal of commonality of functions among the systems surveyed especially in regard to commands relating to program and file management.

2. There were a number of functions contained in the systems surveyed which were not reflected in the functions list. Apparently earlier versions of the functions list would have been more suitable for the matrix.

2.5 DIVERSITY OF LANGUAGES. Regardless of the commonality of functions demonstrated by the matrix, there was considerable diversity in the language implementation methods of the functions within operating systems both syntactically and lexically. In general the surveys and matrix pointed out that there are a finite number of functions for which an operating system can provide control regardless of its complexity. However, within the industry today there is an almost infinite variety of ways of combining and implementing these functions from a language standpoint.

3. TYPES OF USERS OF COMMAND LANGUAGES

3.1 NON-PROGRAM ORIENTED USERS.

3.1.1 Primitive Users. In this category are grouped all those users of systems who do not have any control over the system. In this sense, they are zero function users or non-users. Nevertheless, they represent by far the largest body of system users. These are the kind exemplified by the watcher of the ticker tape display. He is aware of the system and is cognizant of its output from minute to minute, but does not have any control over what is contained in the display.

Another of this type is the race track goer who watches the Tote Board. Here he will react to the displayed information which is dynamic but will not be able to directly affect the values shown.

3.1.2 Circumscribed Users. In this category are included all those, who although they make inputs to the system and receive outputs, in no way affect the progress of the system. As an example of this category, consider the airlines reservation clerk. The clerk inputs data, receives answers from the system, but nothing the clerk can do will cause the system to alter its processes. In cases like this the users are reacting with the data contained in and controlled by the system rather than the system itself.

Another instance of this category is the payroll clerk. In many cases he feeds the data bank of the system and receives in return summaries and listings showing the current state of his payroll. In no way can he directly affect the operation of the system running the payroll packages, and even may not be allowed to alter the progress of the payroll programs themselves.

A last example of this kind of user is the desk calculator. Here the control that the user exerts is over the application package providing the desk calculator facility and not the operating system. This user as with others in this category is isolated from the operating system which exists behind his application. In fact this type of user is unconcerned with the operating system and its controls and will not be able to tell when the character of the operating system changes, so long as his package of applications still runs.

3.2 PROGRAM ORIENTED USERS.

Under this broad category are included all those who are aware of the operating system. These are the users who have some degree of control over the operating system. They may be distinguished from the preceding categories in that they are somehow program oriented. They think in terms of programming and programming systems, and are to some degree aware of the underlying hardware. This general group has some well defined sub-categories.

3.2.1 General Use. This class of user is characterized by the fact that he does a wide variety of operations on the operating systems. He uses several of the programming facilities including the high level languages to achieve

solution to his problems. He may also use application packages for specific purposes. In general, however, he may be categorized as applying the resources of the operating system to solve his problems and arrive at answers to his questions.

3.2.2 Application Programmer. This category of user may use more than one application package as well as almost all high level languages to construct new programming sub-systems to be used by circumscribed users. He makes free use of the control language of the system to create these packages. The only real distinction between this category and the general use category is that this user is an indirect one. This user is not writing programs to solve his own problems but to construct the tools for the solution of someone else's problems. It is expected that this category of user may be more sophisticated than those in the general category. It may be the case that certain parts of the control of the system is permitted to him and not to the general users.

3.2.3 Operations Use. This is a broad category of user engaged primarily in keeping the operating system running for others. Primary among these is the familiar system operator. The traditional concept of the system operator is being replaced in part by a whole range of specialists. Among these are the systems administrator, the Data Base Manager, the Security Officer, and others. This category is characterized by the fact that these users are not solving problems by using the operating system. They do not tend to use high level languages to any great extent. They do tend to modify the system to provide better or more reliable service. They also tend to be the principal people engaged with the physical handling necessary around the outsides of the system. As a general rule, many of the functions performed by this category of users is quite distinct from other program oriented categories. Yet frequently these users tend to 'wear two hats' and only perform these operations functions part of the time. At other times they are quite likely to be general users.

3.2.4 Systems Programming. Although in many senses this category is a logical extension of the applications programmer, it is separated in this case because there is a unique element to this category. These are the users who are expected to modify the operating system itself. Included in this category is the classical systems programmer who constructs operating systems or parts of them. Also included is the system maintenance user, the trouble shooter, the system debugger or fixer. These users, although they do everything that an applications programmer might do, have additional power to affect the system at a more profound level. The access these users have to the insides of the operating systems makes them at once both very dangerous to the reliability of the system and at the same time essential to its continued existence.

Another reason for separating these users from all others is that they must to some degree be dependent on the internal details of the operating system and its underlying hardware. In this, they can not be entirely system independent. This category of user is probably so close to the system that their functions are shaped by the system they deal with.

4. HARDWARE CONSIDERATIONS

4.1 HARDWARE STUDIES. Although considerable discussion of the hardware implications of OSCL was undertaken, very little concrete was realized. The committee did undertake to recommend a 'break-in key' standard and the balloting and final text of that recommendation is included in Section 6.

4.2 WHAT REMAINS TO BE STUDIED IN THIS AREA. It was the opinion of the committee that further work in investigating the impact of new hardware upon OSCL as well as the impact of OSCL upon existing hardware must be undertaken. The committee did not have the time nor the expertise to undertake such study in depth. It is not clear that such a study can ever be completed. New hardware devices will continue to appear and each of these may have an impact on OSCL.

One of the areas of particular concern is new media as input for OSCL. In the past most commands have been entered by card readers and typewriters. It is certain that new media such as displays and audio input devices will have some effect on command structure.

5. RECOMMENDATIONS

5.1 NEED FOR A STANDARD. Although many different programming languages exist for the legitimate purpose of writing computer solutions to problems in a great diversity of application areas, it is not reasonable to expect that a large number of control languages need exist. The purpose of OSCL is single, to control the operations of the system. The advantages of being able to move from system to system without having to relearn control language is akin to the ability to use COBOL on any system without having to learn a completely new langauge.

The need for a standard is pressing and its attainment should be possible. In order to get to the standard, work should proceed with the necessary preliminaries.

5.2 SCOPE OF OPERATING SYSTEM CONTROL FUNCTIONS. It is the feeling of the committee that there are several capabilities which are not necessarily germane to the executive control of a computer system. We recommend that the study group should examine but not feel constrained to include the following features as part of OSCL:

1. editing of documents, programs, or OSCL procedures,
2. formatting of output data or control of output devices, and
3. data base creation, manipulation, or interrogation.

5.3 NO CANDIDATE. There appears to be no candidate among surveyed OSCL's to be selected as the basis for the standard. All the languages have good and bad features. Therefore, we are not recommending a candidate language.

5.4 NO MULTIPLE STANDARDS. We urgently recommend a single standard for OSCL. Any user once having mastered the necessary knowledge of OSCL should not be required to learn another language.

As far as can be determined, the only possible excuse for multiple standards might occur if the differences in user media (i.e., decks of cards vs. scopes) might make it impossible to achieve a single standard.

5.5 PIECEMEAL STANDARDS. To minimize problems of conversion and transition, every attempt should be made to avoid piecemeal standardization.