THE CAIS 2 PROJECT

Sue LeGrand
SofTech, Houston, Texas

and

Richard Thall
SofTech, Waltham, Massachusetts

ABSTRACT

The Common APSE Interface Set (CAIS) is a proposed MIL-STD intended to promote the portability of Ada Programming Support Environment (APSE) tools written in Ada. The standardized interfaces define a virtual operating system, from which portable tools derive their basic services, e.g., file management, input/output, communications, and process control. In the Ada world, such a virtual operating system is called a Kernel Ada Programming Support Environment (KAPSE). The CAIS is a standardized interface between KAPSEs and tools. The CAIS has been proposed as a starting point for standard interfaces to be used in the NASA Software Support Environment (SSE) for the Space Station Program. This paper describes the status of the CAIS standardization effort and plans for further development.

BACKGROUND

The proposed standard [1] was prepared, largely, by a volunteer group composed of members of the KAPSE Interface Team (KIT) and KAPSE Interface Team from Industry and Academia (KITIA). The KIT/KITIA is composed of 60 representatives from U.S. government, industry, and universities, as well as foreign governments and institutions. One seat on the KIT has recently been created for a NASA representative. A small core group of dedicated KIT/KITIA volunteers was responsible for producing the proposed CAIS 1 standard issued in January 1985. Public review of CAIS 1 is now being completed as part of the normal military standardization process. Unless insurmountable objections are recorded, CAIS 1 will become a MIL-STD. A number of prototype implementations of CAIS 1 have been or are being constructed for experimentation and validation of the design. No significant test of the design with tools that use the interfaces has yet been undertaken.

Design of a comprehensive interface set is a large, complex problem. Since the resources of the original volunteer group were severely constrained, CAIS 1 effort was focused on the problem of defining the major structural elements of the virtual interfaces, i.e., the data structuring model, the process control model, and input/output. Many subjects could not be addressed in the requisite time. These include configuration management issues, device control, resource management, issues related to distribution, interfaces between Ada tools, data exchange between environments, data typing in the file structure, and graphically oriented input/output.

In late 1985, a contract was awarded to SofTech, Inc. of Waltham, Massachusetts for the continued development of the CAIS. The enhanced CAIS is called CAIS 2. Compusec, Inc. of San Diego, California is a consultant to SofTech for issues relating to multilevel-secure operating systems and access control. The Naval Ocean Systems Center (NOSC) in San Diego, California, is the contracting organization acting in behalf of the Ada Joint Program Office. NOSC provides the technical lead for all KIT/KITIA and CAIS-related programs.

CAIS 2 DEVELOPMENT

The primary goals of the CAIS 2 project are to produce a standard that:

- meets practical requirements,
- is technically superior,
- is developed with responsive public review, and
- has adequate supporting material.

The major products of this project are a draft CAIS 2 Standard and a final CAIS 2 Standard. These are currently slated for publication in early 1987 and 1988, respectively. Experience has shown that it is exceedingly difficult to understand a software interface standard in the absence of considerable supporting documentation as well as an operating model. For this reason, CAIS 2 will be accompanied by a Rationale, a Guide for CAIS Implementors', a Formal Semantic Description of CAIS 2, and a prototype. Rationale documents will be published with the draft and final CAIS 2. Other supporting items will become available in the year following the publication of the final CAIS 2 Standard.

Public review meetings are planned after the publication of the draft and final CAIS 2 Standards as one method for obtaining constructive criticism from a wide audience. A more formal comment mechanism will also be available during these periods. At other times, the KIT/KITIA acts as the sounding board for design issues. As a guide for CAIS 2 design, the KIT/KITIA has published requirements for CAIS 2 [2]. These requirements are also subject to public review and comment.

CAIS 2 REQUIREMENTS

A few of the major CAIS 2 requirements are paraphrased below with commentary relating to NASA issues.

General Requirements

The CAIS services are intended to be sufficient to support tools used for software development. There are no requirements for real-time services as might be required by many NASA applications. Except for some aspects of communications, software development has no time-critical component. Support for testing of applications which have time-critical features is not addressed by the requirements.

The CAIS shall be independent of any specific operating system or computer. However, a reasonable level of modernity is assumed.

When implemented with sufficiently sophisticated hardware and software, the CAIS shall be capable of supporting multilevel secure operations. In other words, CAIS access control mechanisms must be sufficiently robust to provide for the partitioning of data, users, and devices which are commingled in a common system, but operating with differing levels of security clearance. Some data and devices will be shared, others must not be. This requirement is critical for Space Station operations where classified military and proprietary industrial applications must all share a common facility.

The CAIS shall incorporate existing standards to the greatest extent possible.

The CAIS shall be designed to allow tools to operate in distributed environments. The least constrained model of distribution is a network of computers, each having independent memory and file storage. The database can be shared among the nodes of the network. Computers in the network may be of the differing types. This model should be sufficient to support the SSE.

Data Base Requirements

The requirements mandate support for a sophisticated file system, very close to what is usually called a database management system (DBMS). The DBMS is to support a very general structuring capability, e.g. an entity-relationship model.

No specific configuration management capability is required; although it is tacitly assumed that the structuring capability be general enough to support almost any configuration management method.

A database typing mechanism is required to control the name space of the data objects, the attributes possessed by data objects, and the nature of relationships that can be created among objects.

Robust access control is required. In addition to the conventional discretionary access control, mandatory access control for multi-level classified material is required.

CAIS 2 is required to supply a mechanism by which certain database operations trigger the execution of user-defined procedures.

CAIS 2 will supply a means for grouping database operations into transactions. When transactions are used, the database is permanently modified only when an entire transaction succeeds. Failing transactions result in no change to the database. It is also a requirement that the effect of running transactions concurrently shall be the same as running them in some serial order.

CAIS 2 is required to supply a mechanism for collecting and storing information about how database objects were generated. For example, the history of an object module would include the names and revision numbers of all source files used in the compilation, the name and revision number of the compiler used, and the parameters given to the compiler.

A standard data interchange format is required.

Program Execution Requirements

One executing program can start, stop, suspend, and resume other processes to which it has access.

The CAIS will supply a means for interprocess synchronization and communication.

The CAIS will supply a means whereby one process can monitor the execution of another process. This is useful for debuggers and other dynamic analysis tools.

CAIS 2 PLANS

The design of CAIS 2 has progressed to the point where some general statements can be made about CAIS 2 and its relationship with CAIS 1. We expect CAIS 2 to address all issues explicityly deferred by the CAIS 1 team. We expect simplifications in some areas. However, since the scope of CAIS 2 is significantly larger than CAIS 1, the overall complexity level may be similar. The issue of inter-tool interfaces will be addressed by proposed standard representations for textual and graphical data. CAIS 2 designers do not believe that standardization of inter-tool interfaces more specific than these are within the purview of the effort.

CAIS 2 will not alter the basic structure of the CAIS 1 database model. We expect, however, to conceptually simplify the discretionary access control mechanism. A typing mechanism will be superimposed upon the present entity-relationship model. This mechanism will allow new types of objects to be created by reference to existing types. There will be one base type for all objects, so that tools which operate on all database objects will not be affected by the creation of new types of objects. As a minimum, the typing mechanism will manage the name space of database objects as well as the allowed attributes and relationships. It is not clear if the typing mechanism will deal with the representation of database objects. A few additions to CAIS 1 database services are expected for support of distributed databases.

CAIS 2 will maintain the present process model, i.e. tree structured process creation with a few embellishments. It is likely that some changes will be needed in the area of interprocess communication and control in order to support distributed environments.

The entire input/output model of CAIS 1 will be streamlined. The present model has built-in services for specific classes of devices, e.g. scrolling terminals, page terminals, and form terminals. Not only does this approach proliferate the number of interfaces, but it fails to promote the notion of device independence. For example, given a tool written for a page terminal, it could be difficult to redirect output from that tool to a scrolling terminal. While it may not be possible to achieve satisfactory operation of a screen editor from a Model 33 Teletype, we do not want the interfaces to encourage the construction of device-dependent tools. To accommodate differing devices, we intend to propose the notion of a logical device driver (LDD). An LDD is a program fragment that converts information in a standard representation to and from a stream of commands for a known device, producing the best rendering possible, given the device constraints. Under this proposal, CAIS 2 will have specific interfaces for LDDs, in addition to the normal tool interfaces. If the LDD interfaces can be defined with the correct blend of flexibility and specificity, it should be possible to write LDDs in Ada and transport them from one CAIS implementation to another. In other words, tools would be largely device independent but would depend upon a specific collection of device drivers. Tools would be ported with their associated LDDs, if the LDDs are not already present on the new host. New devices would require new LDDs to be created; however, a new LDD should allow most existing tools to be used with the new device, unless the tool or the device has some unusual characteristic. The LDD concept would allow CAIS 2 implementations to utilize new devices without circumventing the Standard, or necessitating a change to the Standard.

Like CAIS 1, CAIS 2 will supply a bridge to I/O facilities defined by Chapter 14 of the Ada language standard. These facilities are sufficient for a large number of tools, many of which will exist prior to or outside of CAIS 2 implementations. This bridge will allow such tools to be imported into CAIS 2 environments with minimal source code alteration.

We have proposed that CAIS 2 define a few standard data representations sufficient for a large proportion of tools. One representation would be used for sophisticated text. It would encompass the conventional ASCII

character stream, but augment it to support multi-font, multi-format, multi-color realizations. This representation would be bipartite, separating the text stream from the description of how the stream is to be displayed. We have also proposed a standard representation for graphical images. This representation would subsume the sophisticated text representation. Finally, we have proposed a standard language for describing how physical file layout corresponds to the Ada file specification. This would allow files to be converted so that data can be moved across the boundaries between computers, operating systems, KAPSEs, and compiling systems. A standard interchange representation will complete the capability for moving data between CAIS implementations. This capability is key to the success of heterogeneous distributed systems such as SSE.

CAIS 2 designers hope to be able to apply the concepts of standard representations and LDDs to other areas of the CAIS in order to build some resilience into the Standard. A standard as comprehensive as CAIS cannot survive unless it can be rapidly adapted to changing hardware technology as well as the demands of sophisticated applications such as the Space Station.

REFERENCES

[1] Military Standard Common APSE Interface Set (CAIS); 31 January 1985; Department of Defense, Washington, D.C. 20302.

[2] DoD Requirements and Design Criteria for the Common APSE Interface Set (CAIS); 13 September 1985; Ada Joint Program Office, Washington, D.C.