Technical Report
CMU/SEI-90-TR-14
ESD-TR-90-215

CASE Tool Integration and Standardization

Paul F. Zarrella

December 1990

Technical Report
CMU/SEI-90-TR-14
ESD-TR-90-215
December 1990

CASE Tool Integration and Standardization

Paul F. Zarrella

CASE Technology Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

(signature on file)

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Research Access, Inc., 800 Vinial Street, Pittsburgh, PA 15212.
Phone: 1-800-685-6510. FAX: (412) 321-2994. RAI also maintains a World Wide Web home page. The URL is http://www.rai.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145. Phone: (703) 274-7633.

Table of Contents

1 Introduction 1

2 Integration Issues 5
2.1 Single-Vendor Tool Integration 5
2.1.1 Problems 5
2.1.2 Resolutions 6
2.2 Multiple-Vendor Tool Integration 7
2.2.1 Problems 7
2.2.2 Resolutions 8
2.3 Operating Environment Integration 9
2.3.1 Problems 9
2.3.2 Resolutions 9
2.4 Development Process Integration 10
2.4.1 Problems 11
2.4.2 Resolutions 13
2.5 End-User Integration 15
2.5.1 Problems 15
2.5.2 Resolutions 16

3 Standardization Issues 17

4 Standards Efforts 19

5 Outlook/Conclusions 23

Glossary 25

References 29

List of Figures

Figure 1-1 Levels of CASE Integrations 2
Figure 2-1 Types of Integration 12
Figure 2-2 Full IPSE Model 14

CASE Tool Integration and Standardization

Abstract: CASE tool users are faced with the task of coordinating tools and data from a variety of sources spanning the entire software development life cycle.Despite much discussion and increased standardization activity, complete, transparent CASE tool integration is still a long way from realization. There are a number of factors which have complicated the tool integration scenario and a number of actions being taken in an attempt to resolve the problems. The implications of these concerns can be examined from the perspectives of single-vendor, multiple-vendor, operating environment, development process, and end-user integration. In addition to specific technical and methodological solutions, standards efforts are viewed as a possible path to tool integration. To date, formal efforts have done little to resolve the integration problems, but de facto standards may well become the cornerstone of future CASE tool evolution.

1 Introduction

On the surface, Computer Aided Software (Systems) Engineering (CASE) appears to have the potential to improve software development productivity, reduce software maintenance costs, and enhance overall product quality. Some tools offer automation services covering all phases of the development life cycle, including analysis and design, code generation and version control functions. Despite the existence of numerous tools which facilitate development, a major impediment to realization of a comprehensive solution is the current state of tool integration.

One single term that appears in most (if not all) descriptions, discussions, and reviews of CASE tools, is the word “integrated.” There are major differences in interpretation of this seemingly simple term. CASE tools are referred to as integrated simply because they share a similar user interface or because a vendor offers several tools to support more than one phase of the software development life cycle. Some speak of CASE tool integration in terms of the support of shared data storage [26]; others describe a fully supported development life cycle [17] where all tools interface to a common framework/database in a distributed environment [22].

Analysis of CASE tool integration is generally separated into three functional areas: data, control, and presentation integration (see Figure 1-1). Along these lines, CASE tools cannot yet be defined as fully integrated. In a limited sense, most CASE tools/toolsets can be defined as data integrated (or “joined”) with an underlying design database. The tools may share this “dictionary,” but may be limited by any level of scope, interoperation, or environmental capabilities. These tools may not represent or interact with the multiple representations or phases of objects required in coordinated, heterogeneous data storage. Most tools do not support the external control interfaces and formats that are required for automatic inter-tool process/data flow.

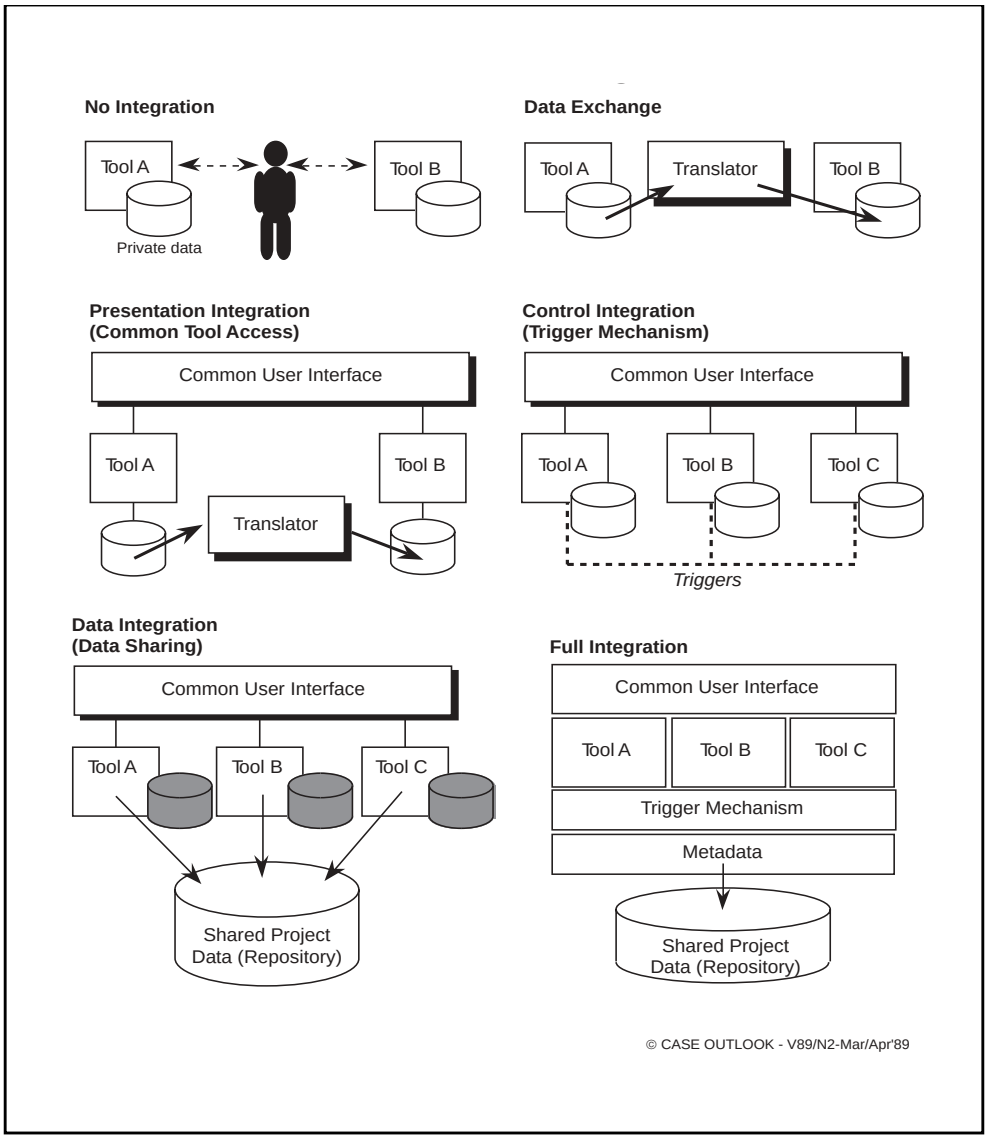


Figure 1-1 Levels of CASE Integrations

Part of the integration roadblock stems from the lack of a concise set of broadly accepted CASE tool standards. Each vendor has invested significant amounts of time and money in the development of proprietary tool interfaces. Each would like to provide the basis of industry standard. Realistically, vendors want to expend the least amount of effort necessary to adhere to any finalized standard without supporting multiple conflicting standards. To this end, vendors are examining alternative approaches to standardization, including forming vendor partnerships and lobbying for acceptance of specific standards.

CASE vendors might come to agreement on standards if there was specific indication of what CASE users wanted/needed from the technology. This would be aided by accumulating user

feedback detailing findings from the adoption/usage of CASE tools. Unfortunately, the (limited) available results from CASE tool adoptions provide contradictory information [21]. One of the main reasons that users aren't adopting CASE tools in the numbers necessary to influence standardization is because of the lack of integration/standards [10], [16]. Given this dichotomy, emerging standards will either be determined by default, or will continue to be dynamically re-defined. Each of these approaches to standardization presents its own set of problems.

One thing is clear: different levels of product integration and standardization affect a user's decision whether or not to adopt and use CASE tools. This report addresses the issues, problems, and resolution efforts related to CASE tool integration and standardization from the users' perspective.

2 Integration Issues

As previously mentioned, there are different viewpoints on what constitutes CASE tool integration and from what perspective CASE tool integration should be considered. With this in mind, the discussion of tool integration will be approached from several different aspects relative to the perspective of the end-user. Five areas of integration are examined here, namely:

- Single-vendor tool integration
- Multiple-vendor tool integration-
- Operating environment integration
- Development process integration
- End-user integration

2.1 Single-Vendor Tool Integration

The initial form of CASE tool integration is “internal” integration, that is, integration of the tools and data of a single vendor. Some vendor tools use a local data dictionary; others fashion a toolset joined together around a central, shared dictionary. The data dictionary is usually some form of (relational) database which offers a vendor a way to provide reliable data storage and access for the tools. This form of integration is generally of a proprietary nature [1].

2.1.1 Problems

A single-vendor approach limits users to the range of tools afforded by the particular vendor. Most CASE vendors do not offer a full life-cycle complement of (integrated) tools, nor do they produce tools for both the personal computer and workstation/mainframe environments. When a single vendor offers a set of tools, the user may be constrained by varying degrees of usefulness/value derived from the different tools. Many users find it unacceptable to be restricted to a single-vendor CASE tool solution[16, 18] due (at least in part) to these limitations.

Some tools, even though from the same vendor, use localized (non-distributed) data dictionaries to store tool objects. This creates problems with data content consistency between tools and users and puts the burden of data synchronization and reconciliation on the user. Also, end-user tools are often precluded from interfacing with the dictionary directly as most data dictionaries are proprietary in nature, use unpublished proprietary interfaces or use incompatible or proprietary database technologies.

The single-vendor aspect also impacts the amount of flexibility or modification in a toolset from which a user can benefit. If a vendor allows a user to modify the interfaces/objects of such a nonstandard toolset, it could very well further widen the “compatibility gap” with other (vendor) tools. Even if conversion capabilities are developed to allow a database/interface to be adapt-

ed to a standard format, the ultimate responsibility for adapting the modifications may be left to the user.

The use of relational databases as the basis of a CASE data dictionary [2, 4] is also problematic. Relational databases cannot readily accommodate the flexible, complex object/data types (e.g., code segments, design diagrams, user processes) required by CASE technology. This is due to limitations of the relational model wherein the database is generally unable to handle the amount of data that it takes to implement a fine-grained object management system (i.e., objects are composed of data items and interrelationships that are more complex than the level afforded by a singular file or character string representation). The problem here is actually a function of data size relative to performance characteristics. Since relational databases cluster data by type rather than by the relationships within the data, the large amount of data required by CASE greatly increases the amount of time required to process database transactions.

2.1.2 Resolutions

Some vendors have merged in an effort to acquire leverage from each other's tool offerings and to fill in the technology/product gaps [20]. This solution changes the scope of the problems encountered in the original single-vendor situation, but eventually it, too, fails to solve the problem. In most cases, upper limits or gaps exist where the product sets remain lacking.

Some vendors are expanding on the concept of shared data dictionaries (or "encyclopedias"). This will help alleviate the problem of database reconciliation, although, if not carefully implemented, it could negatively affect the performance profile of the toolset. Multiple (distributed) tools/users attempting concurrent access to a shared dictionary could exceed the limitations of the development environment. Shared toolset databases, while not explicitly solving the single-vendor limitation, provide a common basis for internal tool integration and a means of transition to external integration.

Efforts are currently underway to expand the integration of object-oriented databases (OODB) with CASE tools. An OODB is the basis of next-generation "repository" products currently under development by both IBM and DEC (see Section 2.2). OODBs differ from relational databases by storing data maintenance and access rules along with the object data. This technology would provide extended capabilities (e.g., multiple object views, modifiable rules, and object types) and would enhance the distribution/performance aspects of a shared data dictionary (or repository) [2] (see discussion of Integrated Project Support Environment (IPSE) below). Universal acceptance of OODB technology is impeded by the fact that the technology is relatively new (no single data model) and by the prior commitment to relational databases of many vendors, users, and standards organizations.

2.2 Multiple-Vendor Tool Integration

Another form of integration is “external” integration. This is when a vendor integrates tools with those of another vendor. This integration can be of the form of “access control” and/or “data control.” In access control, the vendor allows the tools to be invoked/controlled by tools from other vendors and returns appropriate messages/codes to the invoking process. In data control, the vendor allows these external tools to (in)directly manipulate the data contained in the internal dictionary. This section focuses primarily on the data control aspect of external integration. Access control is discussed as part of the “Process Integration” section below.

2.2.1 Problems

Many vendors realize the importance of publishing their tool/data interfaces to promote greater acceptance and usage by the CASE community [11, 26]. This helps other vendors (and users) integrate their own tools with the vendor tools, but does not address the larger problem of data incompatibility as the meaning associated with the data is still not fully described. Not only might the format of the data be different, but the contents of the dictionaries may be inconsistent. The weaknesses of relational databases are exacerbated when objects are transformed. Since data rules are imposed by the tools (as opposed to the database), the consistency of objects (the “view”) may be distorted as semantic meaning is lost or misinterpreted when moving the data between tools.

Some vendors have developed (or are developing) import/export utilities and tool extensions to deal with the problems of data dictionaries that cannot be shared by different tools or among multiple users (e.g., IBM (External Source Format), Interactive Development Environments (IDE) (Visible Connections), Cadre (Teamwork/ACCESS)). However, there is an inherent problem of data loss/mismatch when attempting to merge two (possibly) incompatible data sets. Insufficient data may be either exported or imported. Data objects/relationships used by the exporting tool may be extraneous to the importing tool or may not fit into its data model (e.g., type, format, naming conventions).

Another problem with external integration directly underlines the importance of the selection and acceptance of standards. Vendor integration of the tool/data interfaces of another vendor is a very costly proposition. The cost ultimately includes the indirect support and maintenance associated with the tools and interfaces of another vendor. The costs are multiplied exponentially by the number of nonstandard interfaces and the number of platforms and environments supported by these other vendors. It is highly unlikely that smaller CASE vendors will be able to continue to compete in this type of environment for very long.

Ultimately, given the current state of standardization efforts, end-users may have to act as systems integrators and write the code necessary to effectively integrate a set of vendor tools. This, of course, assumes that the tool interfaces are documented by the vendors and that the user can expend the time/cost necessary to implement the tool integration. In the interim, users are left to manually coordinate changes between tool databases.

2.2.2 Resolutions

Work is underway by IBM, DEC, and others to define a central data repository that would take steps to solving the data incompatibility issue [3, 27]. A repository is a common shared database that stores the rules (or relationships) associated with CASE data objects. The data itself may reside in the repository or may be contained in local databases distributed throughout an application network. Many tool vendors are currently lining up to support the repository concept [6, 14] although they are still maintaining their own separate proprietary product lines as an insurance measure. The repository approach, even as a de facto standard, allows users to effectively “mix and match” tools from different cooperating vendors in order to choose the most appropriate tools for their specific process.

The IBM repository, defined as an integral part of the IBM Application Development/Cycle (AD/Cycle) environment, could help advance the ranks of CASE tool users [10, 14]. Not all users may be willing to immediately fall in line with the IBM solution as it will (initially, at least) run only on IBM platforms and could require a significant start-up expense [3, 27]. Also, the IBM methodology may be incompatible with some existing CASE implementations; users may not be interested in investing their CASE efforts in a proprietary solution. Finally, there may be problems caused by the timing and delays associated with the IBM product. Some users are already too far into multi-year projects to allow them to reorganize their data to fit with a DB2 based repository in the near term. It is unclear whether or not IBM will offer a repository importation facility.

The DEC repository approach, VAX Common Data Dictionary/Plus (CDD/Plus), is not the clear choice either, although DEC is considered to be further advanced toward the repository model than IBM [27]. DEC has a working version of the repository but is constrained by problems similar to those of IBM. Since the DEC repository is currently offered only under VMS, users are locked into a DEC platform offering for central repository management. Also, CDD/Plus uses yet another proprietary interface which discourages massive third party acceptance although DEC, like IBM, relies heavily on such vendor partnerships.

In the meantime, some vendors have formed (or are forming) strategic partnerships, mergers, or acquisitions to complement their toolsets (e.g., Mentor Graphics and Microtec Research (integration of development tools into CASE environment); Cadre Technologies and MicroCASE (expansion of CASE tool product functions); Transform Logic and Nastec (merger of complementary product lines)). These vendors are teaming up to add to their existing tool offerings and, in some cases, to attempt to indirectly effect a CASE tool standard. As yet, no one partnership has had a significant impact on the standards process; this solution simply transforms the problems encountered by the CASE user in a single-vendor situation into the problems of a two (or three) vendor situation.

2.3 Operating Environment Integration

When discussing CASE tool integration, the operating environment should also be considered. This includes both the base computing environment and the add-on tools and utilities that compose the development support environment. Some CASE tools have versions that run on a personal computer (e.g., Excelerator (Index Technology), Teamwork (Cadre Technologies)), while others are targeted to workstations or mainframes (e.g., Software through Pictures (StP) (IDE), Procase C Environment (Procase Corporation)). Tools are also developed for use on a specific target operating system. The choice of system generally has to do with considerations for the technical (real-time) or commercial (Management Information Systems (MIS)) application to be hosted by the toolset.

2.3.1 Problems

Basically, tools are tied into their environment by virtue of the specific operating system or hardware platform upon which the tools are hosted. The tools may use specific features of the operating system or support toolset that are unavailable on other systems (e.g., windowing system variants, UNIX system variants). This makes it harder (or in some cases impossible) for these tools to be rehosted to other environments. Ultimately, these tools cannot be considered “as is” as part of an integrated offering.

There are also problems associated with scalability when considering a move from one environment to another. Some tools might suffer from performance degradation or from environment limitations (e.g., memory requirements, CPU characteristics, data storage facilities). Tools developed for a single-user PC environment may not support multi-user projects. This problem may be one of concurrence or of project/data size. It may be impossible to run multiple instantiations of the tool, or for the tool to handle the increased amount of data access requests.

Another environmental issue is CASE tool integration with a configuration management system. Due to the nature and extent of the data that needs to be handled by a toolset, no viable CASE offering (regardless of the scope of the toolset) can overlook the importance of maintaining an ordered version control history for data objects. Some CASE tools integrate a simple versioning scheme for file objects, but the concept must be extended to include all data types as object granularity becomes finer than the file level and as the need evolves for control over a set of object types more varied than simple source code. Extension of CASE tools to include configuration support must be given careful consideration so as not to provoke the scalability problems (particularly, data storage and performance limitations) previously outlined.

2.3.2 Resolutions

One possible solution to the environment integration issue is the emergence of a standard for the UNIX operating system. The UNIX system shows promise as a “cross-over” operating system catering to the needs of both the technical and commercial markets [5, 7]. In this respect,

CASE tool vendors would be relieved of the burden of maintaining separate product lines for multiple operating systems by targeting the UNIX system. This would also alleviate many of the rehosting issues facing the vendors as product porting would become an issue of hardware (UNIX system platform) only. Environment transparency and tool/database distribution would be enhanced through the accessibility of existing network support functions (e.g., NFS, TCP/IP).

Sun Microsystems is an obvious proponent of the UNIX operating system. Sun has expanded interest in the CASE market with the introduction of Network Software Environment (NSE), a proprietary configuration management system. Although not actually a CASE tool vendor, Sun is trying to leverage off of the tool/workspace management concepts of NSE by encouraging tool vendors to offer products for the Sun systems as third party offerings.

In addition, DEC has reaffirmed its support for both ULTRIX (DEC's version of the UNIX operating system) and VMS by focusing on the development of tools designed to run in a distributed, heterogeneous environment [25, 27]. This was the basis of the DEC partnership with Atherton Technology aimed at integrating DEC CASE tools and, currently, an object-oriented version of the CDD/Plus repository with the Atherton BackPlane IPSE (see Chapter 2) for the VAX environment [5, 8]. By teaming with Atherton, DEC inherits the integration services of an IPSE and still maintains control over the specifics of the CASE toolset and database. DEC aims to leverage its own platform offerings on the possible acceptance of the Atherton product as a standard for CASE tool integration.

IBM has also entered into an agreement with Atherton to rehost the BackPlane to AIX (IBM's version of the UNIX operating system) [5]. This will, again, help strengthen the position of the UNIX system (as well as the IPSE and, specifically, the Atherton BackPlane) in CASE integration efforts. The agreement provides IBM with an IPSE/repository offering (even if limited to AIX) in the interim while it attempts to consolidate the various components and support platforms of the AD/Cycle environment.

In the support environment area, several vendors of change control software are working with CASE tool vendors to integrate their product lines. Some analysts contend that the object-oriented repositories of future CASE offerings will implement change control implicitly as part of the supporting methodology [15]. This will help to lessen the potential for scalability problems; configuration management functions would simply become a part of the set of object relationships. Regardless of how it is implemented, change control (and impact analysis), extended to all levels of tools/data, will support the rapid modeling/prototyping attributes of CASE.

2.4 Development Process Integration

CASE tools are also working into the framework of the development process. These tools are no longer targeted specifically to the analysis/design phase of software development. CASE tools are being considered to help combine the various phases of the entire life cycle (e.g., project management, analysis and design, configuration management) in anticipation of

smoothing process transitions. Some vendors are (re)examining the feasibility of the IPSE and of Integrated CASE (I-CASE).

IPSE offerings allow for tool data, control, and presentation integration (see Figure 1-1, Figure 2-1). Data integration refers to the coordination of access to the underlying tool database(s). Control integration refers to the coordination of access to the CASE tools, themselves. Presentation integration refers to the coordination of the user interface (discussed further in the "User Integration" section below). The basis of data integration is the repository. The basis of control integration is the software "backplane" (or executive) that provides interfaces to the tools (see). This allows for mixed operating system support and, in some cases, for mixed platform support. In theory, the IPSE allows for data locking and version control via an update transaction mechanism.

I-CASE tools center around a shared encyclopedia. The encyclopedia scales between the dictionary and the repository in terms of complexity. In addition to holding data objects, the encyclopedia stores and maintains the meanings associated with the objects which are then used to derive other objects (e.g., draw diagrams, generate code). In this respect, the encyclopedia is similar to the repository although the encyclopedia, as an extension of a dictionary, does not necessarily store object rules nor is it necessarily designed around object-oriented technology.

2.4.1 Problems

One problem facing process integration is tool flexibility. Users are insisting that tools allow for easy modification and adaptation to their particular process and methods. Most CASE tools remain closed to such customization and those tools that do claim an "open" interface generally allow modification of only the presentation characteristics. In order for CASE tools to be fully integrated, users must be able to modify the characteristic behavior of the tool (e.g., design rules, object types, process), not just the display interface [9]. Users should not have to abide by a strictly imposed process for software development (e.g., waterfall model, spiral model) if they are best served by some internally developed or hybrid process. Also, tools must be capable of adapting to evolutionary changes in the software engineering process/model.

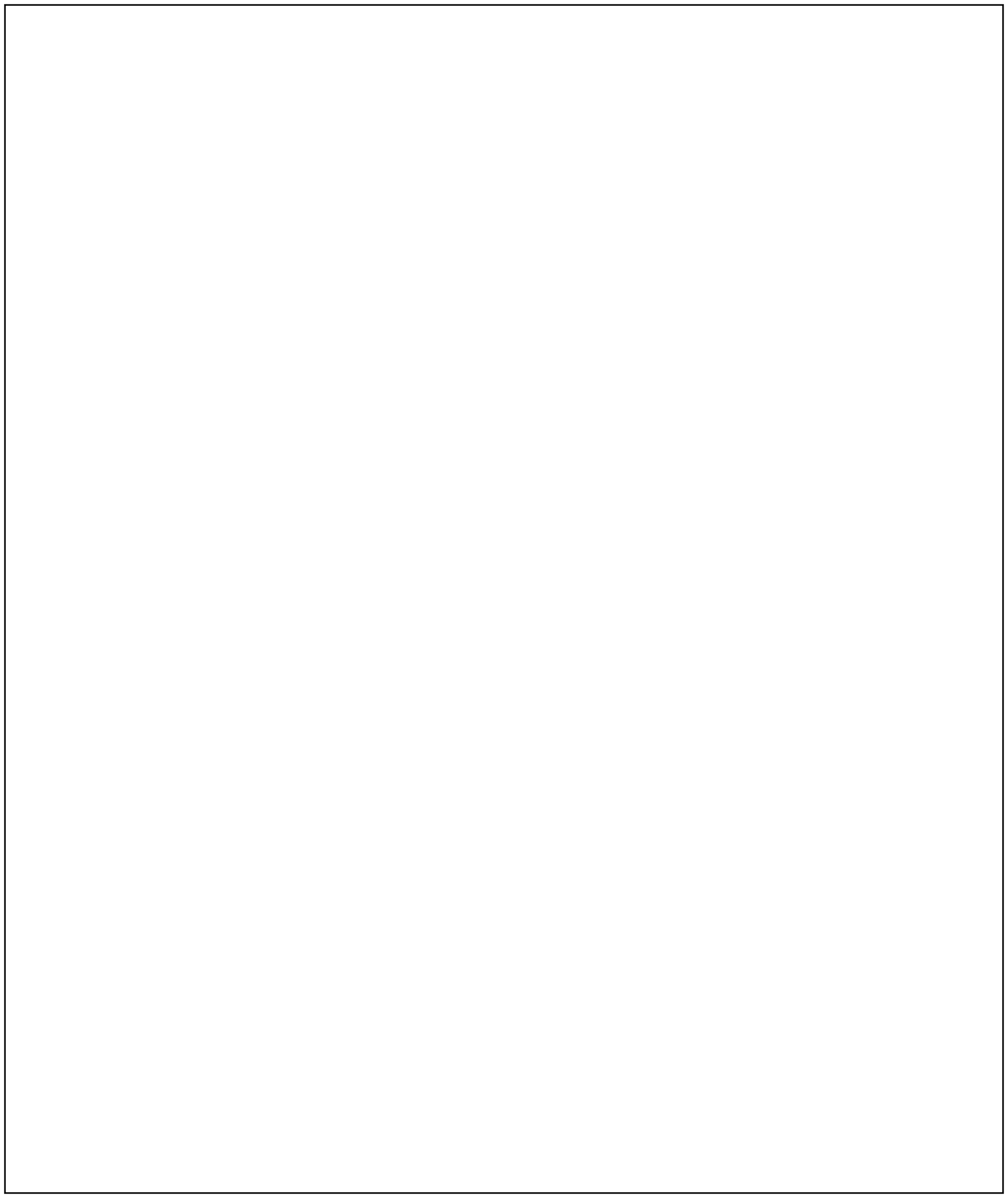


Figure 2-1 Types of Integration

Another problem encountered is that most CASE tools support only very specific design methodologies. Also, the methodology supported by the tools is usually strictly enforced. This requires that users select and learn a new methodology or choose from a limited set of tools that support their current methodology. Users may find that the methodology selected is inappropriate for their organization or that specific methods are replaced in the future by more appropriate methodologies. Also, many users already have some set of methods in place that do not match one of the accepted formal offerings. In the interest of accommodation of such methods, CASE tools need to be able to expand beyond the traditional “bottom up” or “top

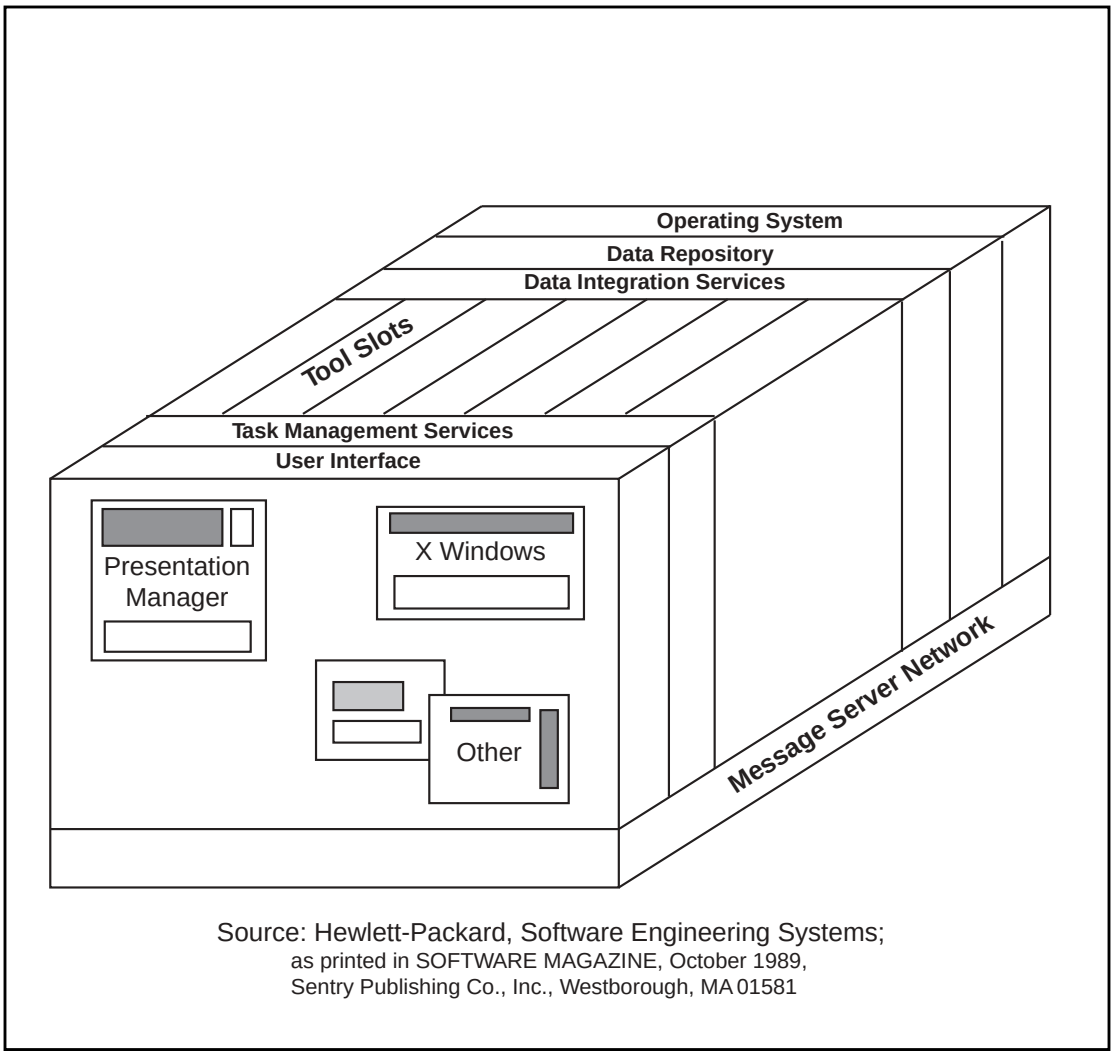
down" design approaches and allow configuration modifications to support alternative ("middle out") methods.

Work on IPSE/I-CASE has been going on for several years. Many vendors see the value of such a tool framework, but few are actively looking to integrate their tools into existing backplanes. Again, the issue of standards comes into play. The IPSE must be tightly integrated with respect to the control interfaces involved in a proper implementation. All of the tools in the IPSE must use standard protocols to support the various control mechanisms (e.g., invocation format, parameter passing, results notification). Most vendors cannot be convinced to expend effort to adhere to an interface that may be superceded in the near future. Since no one vendor can currently define a standard for IPSE/I-CASE integration by offering a comprehensive set of single-vendor CASE tools, no single version of these technologies has been able to claim market dominance.

Users are faced with a similar interface-support dilemma. Some users (e.g., Boeing, EDS, NASA) with an internally developed CASE toolset have gone in the direction of developing their own internal IPSE/backplane variant. The effort required on the part of an end-user to integrate a set of proprietary tools to a vendor backplane could be greater than that required to develop a specific, tailored tool interface. Users may not be willing/able to wait for a standard to emerge or to integrate their tools into a (potentially) nonstandard vendor backplane. By adopting a private solution, the user maintains control over the backplane definition as well as data, interface, and expansion characteristics (i.e., flexibility).

2.4.2 Resolutions

Several comprehensive methodology/toolset offerings are becoming available from large systems/CASE vendors. IBM's AD/Cycle and DEC's CDD/Plus are examples of a combination of methods, central data storage technology, and CASE support tools that are available from (or in support of) such vendors. Each offers the user a consistent set of methods and an outside vendor following that is developing tools to support these methods. Each will also offer the consistency of a central data repository, although neither offering has actually advanced to the point of a fully implemented object-oriented design. Again, the choices presented are mutually exclusive. Each vendor is attempting to define the standard by default, by lining up as many users/vendors behind their products as possible. Ultimately, users may have to live with limited flexibility but may gain from a wider set of tool choices.



Source: Hewlett-Packard, Software Engineering Systems;
as printed in SOFTWARE MAGAZINE, October 1989,
Sentry Publishing Co., Inc., Westborough, MA 01581

Figure 2-2 Full IPSE Model

With the emergence of CASE in the technical market, and the demands of a new set of users, a new emphasis has been focused on the area of IPSEs. Vendors are now looking with renewed interest to incorporation of all aspects of the development life cycle into a seamless product line. Vendors realize that no one company can afford to provide a comprehensive set of CASE tools that spans all of the phases of the life cycle. Cooperation is necessary. So, while not all vendors are supporting all life-cycle phases, most are looking to some form of standardization to define the product interfaces which they must eventually support [9]. In the manner of de facto standardization, IBM, DEC, and Atherton are now likely to end up supporting a common IPSE standard, namely, ATIS (see Section 4). This does not preclude any or all of these vendors from supporting multiple IPSE standards in the future.

Work is also progressing in the area of I-CASE. I-CASE is similar in nature to the IPSE, but is targeted more at the commercial (MIS) marketplace. The emphasis of I-CASE is on the integration (via a shared data encyclopedia) of the front-end phases of the development life cycle that lead to support of code generation (e.g., planning, analysis, design). This is a significant difference as progress on the code generation phase of real-time CASE lags well behind that of the commercial market.

Another term used in the CASE arena is “framework.” Frameworks are essentially extensions of tool backplanes. They provide for the life-cycle integration goals of IPSE/I-CASE via “plug-in” tool integration with a tool management executive. The executive handles the overhead involved with coordination of the tool suite (e.g., user presentation, tool registration and instantiation, error reporting). At the bottom end of the framework is a common data interface/repository, messaging system, and operating system services manager. These interfaces unburden the tool modules of the particulars of the host environment and allow for a broader, more interchangeable product set.

Hewlett-Packard has also been heavily involved in the definition of a framework- type product (SoftBench). The product not only provides a foundation for integration of HP CASE tools, but also provides end-users with an interface protocol (Encapsulator) allowing for simplified integration of proprietary (or third party) tools into the framework [22]. Although a major effort in process integration, the HP framework concentrates on the presentation and control integration and tool messaging aspects of the framework concept while ignoring (at least initially) the issue of data integration (i.e., no common data repository).

2.5 End-User Integration

Finally, more emphasis is being applied to integration of CASE tools with the users themselves. The concept of integration with the end-user ranges from something as simple as maintaining a consistent user interface to something as complex as providing support for an expert system interface to aid in detailed design. In particular, a standardized user interface (also known as presentation integration; see Figure 1-1, Figure 2-1) would help enhance user acceptance of CASE tools.

2.5.1 Problems

One of the basic concerns of integrating CASE tools in general is the issue of a consistent user interface/presentation [9, 22]. The learning curve associated with CASE adoption is very steep and requires a significant investment on the part of the adopting organization. This investment is more than just the cost of the tools. It involves the time and cost of comprehensive training and support (both from the vendor and from the users). Standardized user interfaces and tool function will help keep these costs under control and offer greater tool/choice flexibility to the users.

2.5.2 Resolutions

Some vendors are looking at a standardized user presentation format in order to decrease the learning curve and provide smoother transition between different vendor offerings. Vendors also realize that most users operate in a diverse network of heterogeneous systems and that commonality among these systems is an important consideration in selection of a standard. Some vendors have chosen X-windows as a standard upon which windowing systems for tools presentation are built (e.g., OpenLook (Sun), DECwindows (DEC), Motif (Open Software Foundation (OSF))). This provides a definitive “look and feel” to the diverse toolset offerings and, as found in the HP Encapsulator product, allows the familiar user interface to be extended to user-integrated tools as well. Although no set standard has been officially declared, X-window acceptance is an example of how support by a wide range of vendors (e.g., Sun, DEC, HP) for a particular interface can drive the standards definition process.

In addition to the underlying system, user interface consistency is becoming more of an issue to CASE vendors. There is a general feeling that the user interface contributes significantly to the acceptance and learning time for a product. Since the user interface is the easiest of the integration areas on which to standardize, vendors must use caution not to promote a “quick and dirty” solution. If the vendors do not carefully analyze the requirements of the user interface, without taking into account the context of the tool usage, they may end up with an interface standard that promotes consistency at the expense of ease of use [12].

3 Standardization Issues

As previously mentioned, one of the main obstacles to user acceptance of CASE tools is the lack of approved standards. Users are unwilling to expend a significant amount of effort, time, and expense to adopt CASE tools and then discover that the tools are incompatible with their process or that the tools are ultimately incompatible with the finalized standard. Users are wary of being locked out from CASE technology advances that may be reflected in tools that are incompatible with their own.

In a study done in 1986, it was found that there were approximately 250 tool interconnection standards efforts in progress, with at least 19 of them directed specifically at the CASE arena [19, 24],.Most of these standards are still under consideration and range from government backed efforts (e.g., CAIS-A, SIGMA) to industry efforts (e.g., ATIS, CDIF) to ad hoc standards committees.

One problem facing standardization is the “wait and see” attitude expressed by some CASE tool vendors. These vendors are unwilling to support any specific standards effort or participate in several conflicting efforts in an attempt to “hedge their bets.” In these regards, they reduce the risk that they will adopt the wrong standard and also maintain the possibility that their interfaces already meet or approximate the (yet to be agreed upon) standard. This posture actually detracts from the standards efforts as failure to work actively toward a single standard is, in essence, a position against standardization.

As was previously discussed, there is also the chance of the emergence of de facto standards. Many users and vendors are resigning themselves to the fact that the most immediate possibility for a standard will be the basic acceptance of the technology of one of the major CASE vendors [3, 19]. Irrespective of any bias in the community, in the MIS area at least, this probably means that IBM (and AD/Cycle) will prevail. While there are always inherent problems in this type of standards adoption, at least there would be a focal point on which to build the forward process of enhancing the CASE market. More users would be able to get started on their own CASE adoptions and a wider range of integrated tools would result.

The overall standards effort is further confounded by the fact that different standards would address different integration interfaces. Some standards focus primarily on repository specifications (e.g., IRDS), others on portability (e.g., PCTE), others on data exchange (e.g., CDIF), etc. Many of the standards efforts overlap, but not all members of each standards body belong to all of the standards bodies. Thus, different viewpoints and agendas are driving standards for similar functions which will most probably result in incompatible standards definitions.

The emergence of multiple standards for each of the different areas of tool integration is a very likely possibility. Since most of the standards efforts had been progressing virtually independently of each other (at least until the inception of the International Workshop on CASE (IWOC) Standards Coordination Committee), it could very well come to pass that the first defined standards predominate the industry or that no real consensus emerges at all.

In addition to the problems presented above, there are standards issues that are not specifically addressed by the standards committee(s). The first is the question of metrics. While there are conflicting reports from users concerning productivity increases, no hard metrics can be cited to support the claims. By focusing on this issue, a clearly defined set of metrics could be developed that could be used to quantify tool results. This will help users to measure their progress with respect to both productivity and quality.

Another issue is that of reporting standards. Granted, with the issuance of DOD-STD-2167A, a standard reporting format for military software applications emerged. However, that standard should be addressed to determine if it is applicable to the CASE market and whether or not it complements the activities of the CASE standards efforts. Any standard reporting format should define more than how to print out design diagrams; it should create a framework for extraction of pertinent development cycle information (e.g., usage metrics, configuration management histories, object relationships).

4 Standards Efforts

It would be impractical to list the activities of all of the current standards initiatives or to itemize the charter or specifications of each effort. Instead, several major standards efforts are presented with a brief introduction and summary of the state of the effort.

ATIS/CATIS/CIS

A Tool Integration Standard (ATIS) was originally developed by Atherton Technology to define the interface to the common data repository of the company's Software BackPlane, an IPSE [6, 8]. The "A" in ATIS originally stood for "Atherton," but was replaced after the strategic agreement with DEC and the subsequent changes in the repository model.

ATIS currently defines the object-oriented interface to the DEC CDD/Plus repository. Due to the Atherton agreement with IBM, it is now expected that IBM will participate in the definition of the next version of the interface to include provisions for the AD/Cycle repository. Simply put, the success of the standard is directly related to industry acceptance of some combination of a Atherton-DEC-IBM IPSE model.

The Common Application and Tools Interface Standard (CATIS) is based upon ATIS, but focuses on the properties of an object management system (e.g., object distribution and locking, flexible data types, versioning) rather than on the specific repository interface of ATIS [23]. The divergence is based upon the contention that current database technology cannot effectively support full storage of all object data in a central repository.

The CASE Integration Services (CIS) standard is an off-shoot from CATIS [4]. It supports the object-oriented integration approach, but differs in the control mechanism. While ATIS supports a single point of control (the repository), CIS promotes spreading the control out among several integration services components for data and task management (e.g., message handling, data security, configuration management, repository). This version of the standard had solid industry backing but was endangered by the conflicting interests of its supporters. It now appears that the future of this standards effort is in jeopardy due to the withdrawal of support from several of the key participants (HP, Sun, Cadre, IDE).

CAIS/CAIS-A (MIL-STD-1838/MIL-STD-1838A)

The Common Ada Programming Support Environment (APSE) Interface Set (CAIS) standard was developed by the DoD in response to the interface incompatibility of the independently developed Ada Language System (ALS) and Ada Integrated Environment (AIE) APSEs [2, 30]. The primary focus of the CAIS standards proposal was tool portability. A later version (CAIS-A) added a focus on tool data exchange.

The CAIS-A standard defines a layer that interfaces the tools of the APSE to the Kernel Ada Programming Support Environment (KAPSE) and, ultimately, to the host system. Thus, CAIS-A provides tools with common DBMS and operating system services of the host.

The CAIS-A standard has received lukewarm acceptance by the industry. It is similar in nature to the European PCTE standard (see overview of PCTE below) and will possibly merge with that effort in the future.

CDIF

The CASE Data Interchange Format (CDIF) is a proposed extension (only) to the Electronic Data Interchange Format (EDIF) used for data exchange between CAD tools [4, 13]. This standards effort focuses on the specification of requirements for CASE tool data exchange integration and was originally proposed by Cadre Technologies, Inc.

Although CDIF was originally intended as a quick solution to the data integration problem, a full standard is still years from completion. An Electronic Industries Association (EIA) committee overseeing CDIF expects that a draft of the standard should be available in the first half of 1991.

IEEE (P1175)

This is a tool interconnection standards proposal sponsored by the IEEE [24, 28]. The proposal acknowledges that there is no one constant way to solve the software development problem and that tool advances require users to adopt and mix tools from different sources as part of their toolkit. The proposal advances the concept of better tool integration at the information transfer level.

The proposal addresses the specifications of a Standard Text Language (STL) that describes the information that is transferred between tools. Also, because organizations may undergo a “culture” change (e.g., learning a new methodology, converting information) to use tools, the standard provides a reference model for tool to organization interconnections. And, because the tools require a specific support platform, the standard provides a reference model for tool to platform interconnections.

IEEE Standards Board approval is currently expected in September 1991.

IRDS (ANSI/ISO)

The Information Resource Dictionary Standard (IRDS) focuses on the issue of data integration via a centralized repository [4]. This standard defines the data dictionary interface to a relational database. It specifies four levels to the framework; the definition schema, the data model (object types, attributes and relationships), the database contents, and the underlying database.

Two versions of the standard have emerged: a version that has already been approved by ANSI (1988) and a version (in final draft form) that is promoted by the International Standards Organization (ISO). The ANSI version is based on an entity-relationship data model while the ISO version can support different data models by using a simpler relational view of data based on SQL.

PCTE/PCTE+

The Portable Common Tools Environment (PCTE) standard is a European standards effort aimed at defining the tool support interfaces for an IPSE [2, 29]. PCTE was originally designed to address the issues of interoperability of tools and data between environments. It defines interfaces that provide for a distributed database, a distributed architecture, and an extended, uniform user interface.

PCTE+ was implemented to (among other things) reduce the original constraints of UNIX operating system compatibility required by PCTE; add security, accounting, and versioning extensions; and, refine the object management system so that type definitions are represented as objects.

PCTE is similar in nature to CAIS but differs in that PCTE provides interfaces for tools written in both Ada and C, while CAIS supports only Ada. PCTE has been better received by industry and includes supporters (e.g., European Computer Manufacturers Association (ECMA), NATO) and interested parties (e.g., HP, IBM) from both the commercial and military markets.

SIGMA

The Software Industrialized Generator and Maintenance Aids (SIGMA) standard is a broad-based Japanese effort aimed at all levels of tool integration (e.g., portability, data exchange, repository architecture) [4, 19]. The standard is designed to create a seamless software development environment.

The effort was originally well supported and represented by both Japanese government and industry. A prototype system was completed in 1989 as part of an initial five year development plan, and the effort has now entered its “commercialization” stage. However, there are concerns about future acceptance of the technology. The current tools are lacking in some areas of coverage of the development life cycle and are generally not well integrated. Also, the effort is seen to mainly benefit large mainframe vendors who will now control evolving CASE tool standardization efforts (for SIGMA).

5 Outlook/Conclusions

In the next five years, vendor consolidation and restructuring will bring about some limited improvement in CASE tool integration. For the most part, though, the status quo will prevail. Vendor partnerships will continue to adapt their tools to meet privately agreed upon interfaces and then struggle for elusive industry acceptance. The effects of tool integration and standardization efforts will remain limited. Much discussion will surround each new tool, each new partnership, and each new tool adoption and productivity claim. The bottom line, however, is that the limited integration progress and the conflicting reports about CASE will keep users confused about the future direction of the technology and apprehensive about the selection of a toolset.

In the long term, the standards dilemma itself may bring vendors into alignment over integration issues. IBM (and/or DEC) will most likely force standardization by default in the commercial CASE market. Although it is more difficult to predict what will happen in the less mature technical CASE market, DEC is already firmly established in this area and is poised for expansion. These de facto standards should, at least, cause other vendors to finally come to standards “realization.” That is, as the larger vendors get users to start accepting their products and as these vendors then start touting their particular definition of a standard, other vendors will be forced to join up with the standard or quickly consolidate to accept other (hopefully well defined) standard(s). In this respect, the divergent groups will have to put their differences aside and come to closure or they will be isolated from the CASE mainstream. There is always the possibility that in the interim, one of the vendor partnerships will hit upon an acceptable standard and preempt the larger vendor(s), but this scenario is highly unlikely/unpredictable.

Glossary

AD/Cycle (Application Development/Cycle)
IBM CASE methodology/framework (incorporates common data repository) based on the IBM Systems Application Architecture (SAA) development environment

ATIS (A Tool Integration Standard)
Atherton-DEC[-IBM] CASE standards effort which addresses object-oriented repository interface specification

Backplane
Tool coordination facility providing “plug-in” tool (control) integration with a tool management executive

CAIS[-A] (Common APSE Interface Set)
U.S. DoD CASE standards effort which addresses tool portability and data exchange specifications (similar to PCTE+ standards effort)

CASE (Computer Aided Software (Systems) Engineering)
Collection of tools supporting formalized methods designed to automate and enhance the software development process

CATIS (Common Application and Tools Interface Standard)
CASE standards effort which addresses object management system (repository) architecture specification (derivative of ATIS standards effort)

CDD/Plus (Common Data Dictionary/Plus)
DEC CASE tool data repository developed for VAX environment (incorporates distributed access capabilities)

CDIF (CASE Data Interchange Format)
EIA CASE standards effort which addresses tool data exchange specification (derivative of EDIF CAD standard)

CIS (CASE Integration Services)
CASE standards effort which addresses object-oriented data and task management specifications (derivative of CATIS standards effort)

Commercial CASE
The set of CASE tools/methodologies that support IS/MIS software development efforts (specifically, data-driven tasks)

Configuration Management System
Set of tools that facilitate the collection, maintenance and logical ordering of software module versions and releases

Control Integration
The mechanism for coordinating CASE tool execution and information reporting allowing tools to “trigger” a sequence of pipelined tasks

Data Integration
The mechanism for sharing data objects between/among CASE tools, allowing direct and/or indirect access to local/public data storage facilities

Dictionary
Database for CASE tool data storage (a basic element of all CASE tools), possibly shared by multiple tools

Encyclopedia
Shared database for CASE tool data storage (the basis of I-CASE tool integration)

Flexibility
Customization characteristics of a CASE tool (e.g., modification of user interface, object relationships and/or supported methodology)

Framework
Tool coordination facility providing data, control, and presentation integration for life-cycle CASE tools (similar to IPSE)

I-CASE (Integrated CASE)
Tool coordination facility providing data integration for life-cycle CASE tools in support of code generation (incorporates common data encyclopedia)

IPSE (Integrated Project Support Environment)
Tool coordination facility providing data, control, and presentation integration for life-cycle CASE tools (incorporates common data repository)

IRDS (Information Resource Dictionary Standard)
ANSI/ISO CASE standards effort which addresses repository interface specification

Methodology
Formalized approach used to perform a task (or set of tasks) and to outline the deliverable results (an integral part of CASE technology)

DOD-STD-2167A
U.S. DoD specification/requirements standard directed toward software development and delivery for military applications

NFS (Network File System)
Distributed file system management protocol originally developed at Sun Microsystems (provides client workstations with transparent access to remote server files)

NSE (Network Software Environment)
Sun Microsystems distributed configuration management system (allows data sharing between/among multiple software development projects)

Object Management System
Database of non-redundant project data, providing automatic maintenance of associated object links

OODB (Object-oriented Database)
Database that combines data storage with object relationship/rule processing (basis of advanced CASE tool repositories)

P1175
IEEE CASE standards effort which addresses architecture and tool data exchange specifications

PCTE[+] (Portable Common Tools Environment)
European CASE standards effort which addresses tool portability and data exchange specifications (similar to CAIS-A standards effort)

Platform
Computer system specification incorporating type/version of operating system software and hardware/CPU

Presentation Integration
The mechanism for maintenance of a consistent CASE tool user interface allowing common tool invocation/option functions

Process
The collection of the functional activities related to the various phases of the software development life cycle

Repository
Enhanced (object-oriented) database for CASE tool data storage (incorporates object relationship/rule processing)

Scalability
Ability of a CASE tool to function properly in an extended environment (e.g., number of users, size of project, network distribution)

SIGMA (Software Industrialized Generator and Maintenance Aids)
Japanese CASE standards effort which addresses general (CASE) tool integration specifications

Software Life Cycle
The collection of software development phases representing the full term (from inception to completion) of a project

Spiral Life-Cycle Model
Model of software development life cycle defining process step reiteration and interaction with multiple development phases in a “spiral” configuration

TCP/IP (Transmission Control Protocol/Internet Protocol)
Connection-oriented, end-to-end reliable stream protocol for multi-network applications and the underlying data transmission protocol

Technical CASE
The set of CASE tools/methodologies that support real-time/embedded systems software development efforts (specifically, event-driven tasks)

View
Specific representation or interpretation of data by a CASE tool (e.g., design diagram, document, code segment)

Waterfall Life-Cycle Model
Model of software development life cycle defining process step flow from one phase of development to another in a series of “waterfalls”

X-windows
Computer windowing system originally developed at MIT (potentially a standard basis for CASE tool presentation integration)

References

1 Acly, E. “Looking Beyond CASE.” *IEEE Software*, 5, 2 (Mar 1988), 39-44.

2 Brown, A. W. *Database Support for Software Engineering*. New York: Wiley, 1989.

3 Carlyle, R. “Is Your Data Ready for the Repository?” *Datamation*, 36,1 (Jan 1990), 43-47.

4 Chappell, C., Downes, V., & Tully, T. “Real-Time CASE: The Integration Battle.” *Ovum*, 1989.

5 Cortese, A. “CASE Standard Comes Overseas for UNIX Arena.” *Computerworld*, 24, 2 (Jan 1990), 23-25.

6 Cortese, A. “DEC Challenges IBM CASE Strategy.” *Computerworld*, 23, 41 (Oct 1989), 120.

7 Cureton, B. “The Future of UNIX in the CASE Renaissance.” *IEEE Software*, 5, 2 (Mar 1988).

8 Feuche, M. Atherton, “DEC to Boost CASE Standard.” *MIS Week*, 9, 19 (May 1988), 33.

9 Forte, G. “In Search of the Integrated CASE Environment.” *C/A/S/E Outlook* 89, 2 (1989).

10 Gibson, S. “CASE Buyers Await Repository.” *Computerworld*, 23, 20 (May 1989), 2.

11 Gibson, S. “Some Win, Some Lose When Repository Debuts.” *Computerworld*, 23, 11 (Mar 1989), 141.

12 Grudin, J. “The CASE Against User Interface Consistency.” *Communications of the ACM,*32, 10 (Oct 1989), 1164-174.

13 Hecht, A., & Harris, M. A *CASE Standard Interchange Format: Proposed Extension to EDIF 2 0 0*, Cadre Technologies Inc.

14 Hewett, J. “AD/Cycle.” *Ovum* Software Europe (Feb 1990).

15 Jander, M. Change “Control Meets CASE.” *Computer Decisions,* 20, 10 (Oct 1988), 81-84.

16 Margolis, N. “CASE Fights to Beat ‘All Talk, No Action’ Image.” *Computerworld,* 22, 52 (Dec 1988), 45-48.

17 Martin, J. “Integrated CASE Tools a Must for High-Speed Development.” P*C Week,* 6, 3 (Jan 1990), 78.

18 Mason, J. “Integration of CASE Remains a Distant Dream.” *PC Week*, 6, 6 (Feb 1989), 90.

19 Myers, E. “CASE Standards Connecting: Efforts Worldwide Attempting Coordination.” *Software Magazine,* 9, 12 (Oct 1989), 23-27.

20 Pallatto, J. “CASE Tool Developers Join Forces; Consolidation Seen as Key to Product Integration.” *PC Week*, 6, 30 (Jul 1989), 55.

21 Pallatto, J. “Study Relates Success of CASE Tools; Productivity Increases With Team Use.” *PC Week* 6, 1 (Jan 1989), 43-45.

22 Phillips, B. “A CASE for Working Together.” *ESD: The Electronic System Design Magazine,* 19, 12 (Dec 1989), 55-58.

23 Phillips, B. “Software and CASE.” *Electronic Design*, 37, 1 (Jan 1989), 64-72.

24 Poston, R. M. “Proposed Standard Eases Tool Interconnection.” *IEEE Software*, 6, 6 (Nov 1989), 69-71.

25 Roach, G. T. “CASE: DEC’s View; Part 1: A look at DEC’s CASE Strategy.” *DEC Professional*, 8, 5 (May 1989), 48-54.

26 Wasserman, A. I. “Integration and Standardization Drive CASE Advancements.” *Computer Design*, 27, 22 (Dec, 1988), 86.

27 Yourdon, E. “DEC’s CASE Environment.” *American Programmer*, 3, 1 (Jan. 1990), 4-14.

28 “A Standard Reference Model for Computing System Tool Interconnections.” *IEEE Computer Society Task Force on Professional Computing Tools* (Feb 1990).

29 “Introducing PCTE+.” *Independent European Programme Group* (Apr 1989).

30 Hitchon, C., Judd, M., Pritchett, G., and Thall, R. “Introduction to CAIS; Common Ada Programming Support Environment.” (*APSE) Interface Set (MIL-STD-1838A)* (Sep 1989).