Technical Report
CMU/SEI-91-TR-13
ESD-91-TR-13

Case Studies in Environment Integration

Ed Morris
Peter Feiler
Dennis Smith

December 1991

Technical Report

CMU/SEI-91-TR-13

ESD-91-TR-13

December 1991

Case Studies in Environment Integration

Ed Morris

Peter Feiler

Dennis Smith

CASE Technology/Software Development Environments Projects

Software Engineering Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

Table of Contents

Case Studies in Environment Integration

Abstract: Four environment builders and participants at two workshops were queried concerning the environment standards, implementations, and technology that prove useful in the integration of tools into software engineering environments. Specific information was gathered about the software and hardware environments in which tool integration occurred, the goals of integration, the tools integrated, mechanisms used, and the standards applied. Observations concerning the current state of tool and environment integration are provided, and trends in integration are identified.

1 Introduction

The Navy Next Generation Computer Resources (NGCR) program proposes to establish commercially-based interface, protocol, and services standards for mandatory use in future Navy systems developments beginning in 1995. The NGCR has adopted an open systems approach in an attempt to field more current technology, increase flexibility, control costs, and allow the Navy to benefit from the existing marketplace. Existing NGCR working groups are developing backplane standards (based on IEEE 896 Futurebus+), fiber optic local area networks (based on IEEE 802.5 and ANSI X3T9.5 FDDI) and operating systems standards (based on IEEE 1003 POSIX).

An additional interface area identified as being in need of standardization is that of Project Support Environments (PSE). Initiated during February 1991, the Navy NGCR Project Support Environment Working Group (PSEWG) has as its main objective the identification of a set of industry-based standards to form an "open" framework for project support environment tools, user interfaces, database management systems, and other components of the computing environment. This framework will be applied in the development and maintenance of future Navy systems. Specific objectives include:

- The identification of Navy project support environment requirements, particularly with regard to environment frameworks.
- The identification of existing and evolving environment standards, environment implementations, and environment technology.
- The determination of the applicability of identified environment capabilities to the requirements.
- The recommendation of environment framework interfaces, protocols, and services to the Program Office (SPAWAR-324) for Navy-wide standardization.

Of potential value in the identification of existing environment standards and technology are the experiences of other groups which have built or are building project support environments. These groups possess a wealth of experience regarding approaches and standards selected for projects, as well as information about interfaces which have simplified the building of a PSE from substrates and tools available in the commercial market today.

During the fall of 1990, information for this report was gathered at workshops held at the Software Engineering Institute (the CASE Adoption and the Tool and Configuration Management Integration workshops), through informal contacts with various PSE builders, and by conducting a small number of intensive interviews with organizations involved in building software engineering environments (SEE). Interviews were conducted with representatives of the Boeing Automated Software Engineering Environment (BASE), the RADC funded Software Life Cycle Support Environment (SLCSE), Verdix VADS APSE, and Hewlett Packard SoftBench environment efforts. Information was gathered about the software and hardware environments in which tool integration occurred, the goals of integration, the tools integrated and mechanisms used, and the standards applied.

This report summarizes information gathered from PSE and tool builders. The intent of this report is to identify areas where successful standardization would improve tool and environment integration, as well as to pinpoint areas where successful integration standards exist. The findings of this report will be used by the NGCR PSEWG as background information to help assess the use of current standards and design concepts in the building of a SEE and to help identify the need for additional standards or new design concepts.

Chapter 2 of this report will discuss historical influences on tool support that have helped to create the current situation concerning tools and environments. Chapter 3 presents a rationale for the selection of interview candidates. Chapter 4 summarizes each of the selected environment efforts. Chapter 5 provides observations on the evolution and current state of tools and environments. Chapter 6 discusses these observations in relation to the goals of the NGCR PSEWG. Chapter 7 summarizes and concludes this report.

2 Historical Influences on Tool Support

Software engineering tool technology of the late 1970's was characterized by compiler-centered programming environments executing in a time-shared hardware environment. Tool technology primarily consisted of tightly coupled support for the compilation, linking, and debugging of software. Non-integrated tool support was available for editing of programs and text. Tools were primarily single user, and therefore placed little emphasis on policies to support large-scaled software development.

Two major traditions that developed during the early 1980's have influenced the characteristics of current software engineering environment support. A technology-driven push toward integrated environments offering cradle-to-grave support for software development was the first of these traditions. It spawned many integrated project support environments (IPSE), including ISTAR [1], and PACT [2], and Ada programming support environments (APSE), including Rational [3], and the Ada Language System (ALS). The second tradition, stand-alone commercial tool support for portions of the software life cycle, was essentially a market-driven push to develop new tools to automate additional tasks in the software life cycle. This tradition can be considered the natural extension of tool technology to the next levels beyond simple compiler-centered programming. The resulting explosion of new tools developed in this tradition includes the many commercially available CASE tools and document generation systems.

2.1 Technology-Driven Environments

The IPSE tradition derives from the publication of the Stoneman Report in 1980 [4]. Stoneman described a tool interface (an APSE) that offered services for the building of an integrated set of tools, or an environment. Unlike tools generated via the tools market tradition, early attempts at constructing IPSEs were large-scale, government-supported efforts; however, commercial attempts at developing an IPSE were undertaken [1]. IPSE environments were to span large portions of the life cycle, and be shared by multiple users. The practical constraint of multiple users sharing data required that tools and environments generated by IPSEs provide complex, centralized database support, as well as extensive support for the policies enforcing cooperation.

The nature of long-lived, large-scale software engineering projects addressed by the technology-driven tradition placed many additional requirements on environments. Because the supporting hardware platform could be expected to change over the life of the IPSE and, perhaps, from project to project, IPSEs were expected to be easily portable between multiple platforms. They were expected to provide for plug-compatible tool replacement, and support the tailoring of tool support to individual projects. And they were expected to be suitable for the development of software from many domains.

Due to technical limitations of database support and performance at the time, the construction of IPSEs proved difficult. In fact, research and development of sufficiently capable project databases continues to draw significant interest. Those IPSEs that actually were constructed were not widely adopted.

2.2 Market-Driven Tool Evolution

The tradition embodied in the market-driven explosion of new tools to support software engineering activities was made feasible by the development of low-cost, single-user personal computers and workstations. The graphics capabilities of these computers made it possible to develop tools that automated many of the structured analysis and design techniques pioneered in the 1970’s, as well as documentation tools that could integrate text with graphics.

In their initial manifestation, early CASE tools were market driven to automate popular methods and relied on a simple proprietary database or the file system to maintain tool data. They were primarily single-user tools which only later attempted to provide policies to support multiple developers. Since these tools were built by different companies with little inherent reason to integrate tool actions, control of a set of such tools could at best be decentralized.

As soon as early CASE tools were developed, users discovered that such tools were of limited usefulness, particularly for large projects. Individual tools suffered from a lack of support for decomposition of the tool database, configuration management, and version control. The tools did not provide a mechanism for encoding organizational policies. Tools, including in some cases tools from the same toolset, would not work together. Individual tool databases were inaccessible, interfaces were inconsistent, terminologies conflicted, and data synchronization and reconciliation between tools was often manual.

CASE tool builders spent the early part of the 1980’s learning about and solving the many problems with early tools. By the mid 1980’s, the design and performance problems encountered in developing single-user, graphic workstation-based tools were being solved. Problems of offering support across multiple platforms and providing flexibility to adapt to multiple methods are now being addressed.

2.3 Toward Federated CASE Environments

While early IPSEs suffered through technical limitations and lack of customer acceptance, individual tool vendors constructed first-generation CASE tools which did not rely on IPSE services. As the investment in these individual tools increased, vendor enthusiasm for IPSE standards declined.

Pushed by market forces, CASE vendors did not stand still, but instead attempted to redress many of the initial grievances against CASE tools. In order to make tool use more attractive for large projects, CASE tool vendors added configuration management and version control to products and provided database locking schemes to prevent simultaneous access to tool data. Tools were configured with licensing schemes that allowed them to work in distributed environments.

In order to provide a greater degree of integration with other tools, CASE vendors began providing “open” architectures which define interfaces for other tools which wish to access the tool database. Many vendors have now formed technical and marketing agreements with other vendors to provide a tighter level of integration with other, non-competing tools, such as documentation systems.

While offering a level of tool integration, such environments are not ideal. Because the inclusion of tools in the environment is determined by strategic marketing agreements of the vendors, the customer relinquishes much of the flexibility of tool selection. Since the individual tools remain distinct (and can be purchased separately), the degree of integration between tools is limited. Customers are also limited to the usage models foreseen by the tool integrators. Finally, upgrade paths are limited by the willingness and capabilities of the vendors involved to maintain and improve both the individual tools and the integrated toolset.

As database and interprocess communication technologies have improved, a new concept of an environment framework has been developed by vendors such as Atherton (Atherton Software Backplane) and Hewlett Packard (SoftBench). Like the original IPSEs, such frameworks provide a set of integration services for software tools. However, since they do not require general conformance of the tool to the IPSE format, these services are non-intrusive to the tool. In addition, such frameworks do not attempt to define the policies to be enacted in the environment; rather, they provide additional services to enact policies.

Recently, Wallnau [5] has defined a taxonomy of environment types based on these historical trends in PSE technology. This taxonomy identifies a trend from early, monolithic attempts to build IPSEs, to collections of tool vendors providing limited tool integration (tool coalitions), and finally toward independent tools integrated by an environment framework (tool federations).

Tool federations represent a merging of the traditions of market-driven tools and technology-driven environments. Tools integrated within such federations maintain their own data storage and access mechanisms, as a vestige of the now long history of independent tool development. The federated environment framework provides specialized services to allow tools to communicate actions and intentions, and to share data. The customer maintains a degree of flexibility in choosing which tools are to be integrated in the environment.

3 Rationale for Interview Subjects

Regardless of the specific focus for the construction of an integrated environment, the basic mechanisms used to connect tools have remained remarkably consistent over time. Discussion of tool integration has commonly centered around three major classes of mechanisms for integration of tools: presentation integration (common look and feel among tools), control integration (execution of functions provided by other tools), and data integration (sharing of data).

Tool builders have recognized that presentation integration, control integration, and data integration are necessary, but not sufficient to provide truly integrated environments [6]. The classic forms of integration provide only a mechanistic view, addressing issues such as integrating architectures, interfaces, and techniques. A process-oriented view of integration, which addresses the integration of an organization's development model and life-cycle process with tools and environments has also been postulated.

The specific environments that were chosen for study by the SEI demonstrate the evolution of both mechanistic and process-oriented integration of tools over time. While few in number, we believe that these environments represent a wide range of environment approaches and provide for the formulation of conclusions regarding:

- **Specific mechanisms useful in integration**. Of particular interest are those mechanisms that are universal to all integration efforts, clearly defined, and demonstrated to be effective.
- **The evolution of mechanisms over time**. In particular, we were interested in which integration mechanisms were specific to a time period or particular effort, and which mechanisms transcended all time periods and efforts.
- **The effect of the type of integration effort (in-house, commercial, general purpose, or military-specific efforts) on mechanistic and process-oriented integration**. Often, different efforts can have different goals. For example, while commercial efforts may be aimed at generating short- to intermediate-term sales, military-specific efforts may be aimed at producing an environment capable of long-term support of a software product.

BASE represents an early in-house corporate attempt at tool integration. The BASE effort began in 1983, when Boeing recognized the need for the integration of tools and the encoding of the software development process (process integration) for large DoD projects. Reflecting early tool and environment technology, BASE consists of a loose collection of tools from which data is extracted in batch processing to provide data integration. BASE is unique in that it has undergone major transformations during its lifetime, reflecting parallel transformations in the computer industry from a primarily batch-oriented environment to a more interactive one.

Developed by General Research Corporation under Air Force contract, SLCSE [7] represents a more recent attempt to integrate tools into a toolset useful for the development and maintenance of software subject to MIL-STD-2167A standards. SLCSE implements an entity relationship model of MIL-STD-2167A as a central storage mechanism for data integration.

VADS APSE (Verdix) is a current commercial attempt to wed an existing toolset Verdix Ada Development Systems (or VADS) to a commercial environment framework (Atherton Software Backplane). The VADS APSE effort involves integrating the VADS toolset with the Atherton object-oriented database, configuration management capabilities, user interface, and communications mechanisms to provide data, presentation, and control integration. The VADS APSE effort represents an interesting integration attempt with characteristics of both tool coalitions and tool federations.

The Hewlett Packard SoftBench [8] environment framework represents a unique approach to the integration of tools. Whereas other integration efforts have focused primarily on the integration of data from various tools, the SoftBench product attempts to provide mechanisms for the control of tools working together. The SoftBench product is generating interest in both the CAD (Computer Aided Design) and CASE worlds.

4 Summary of Environments

4.1 BASE

The BASE system has undergone a series of changes in user interface, database, and operating system, as well as in goals and objectives. The original BASE effort was an attempt to build a unique Boeing environment. Later, as the importance of flexibility in the toolset and the environment platform became apparent, BASE builders aimed for portability of the environment between platforms, and plug compatibility of tools. The BASE environment has now entered a maintenance mode. The expense of maintenance is considerable, due to the multiple platforms supported, and the continued release of new versions of tools. As major systems vendors enter the CASE integration market, BASE developers expect their role to become increasingly one of transitioning new tools and environment technologies developed by these vendors to users within the Boeing community.

BASE is a loose collection of commercial off-the-shelf (COTS) and in-house developed tools. Tools in the BASE environment are responsible for maintaining their own data. Data integration is provided by batch processing that extracts data from individual tool databases for storage in a relational database. A Boeing-developed interchange format (BIF) aids data extraction. Data stored in the relational database is later used by document generation and traceability tools developed in-house. Due to Boeing's market clout, vendors worked closely with Boeing to define interfaces for access to vendor tool data.

BASE provides a standard user interface format for in-house and batch-oriented tools. Process integration supporting a Boeing-designed software lifecycle (BSWS-1000) was originally provided by the enforcement of corporate policies in the BASE user interface.

Tool plug compatibility and environment portability have been significant factors in BASE design and maintenance decisions. BASE developers have attempted to use industry standards where they are available to simplify tool replacement. An SQL interface is used for access to the BASE database to simplify replacement of the underlying DBMS. An X-Window/Motif-based user interface has been adopted, replacing the original Boeing-developed system. As BASE has migrated toward different operating environments (MS DOS, VMS, and Unix), a virtual operating system layer was added to simplify porting.

The BASE project encompasses both environment construction and maintenance, and associated transition support. Transition support accounts for a large but variable portion of the BASE budget. The portion of the BASE budget devoted to transition has grown as the need for transition support was recognized within Boeing. Much of the transition budget is devoted toward methodological training for analysis and design techniques.

The pattern of use of the BASE environment varies from project to project. Some projects use the complete BASE environment, while others use only selected portions of the environment. BASE engineers are now involved in the planning of projects at very early stages, including during the proposal stage.

According to BASE project engineers and managers, the primary benefit of the BASE effort is the enhanced understanding of the need for well-defined methods and tools. Even those projects that do not adopt the BASE environment have gained from a growing knowledge base of available tools and methods.

4.2 SLCSE

SLCSE comes closest among surveyed environments to an IPSE-like approach to tool integration, providing data-oriented tool integration and centralized process support. It consists of a large collection of COTS, government-furnished equipment (GFE), public domain, and custom-built tools which vary in degree of independence from the SLCSE data model. Capabilities of individual tools include editing, mail handling, project management, design, development support (compilers, linkers, debuggers), testing support, configuration management, report and document generation, and environment management. Tools provided within the SLCSE environment are intended to support a variety of methodologies and languages throughout the life cycle.

The SLCSE database is centered around an entity relationship implementation of the MIL-STD-2167A life cycle which serves as a repository for project information, and as a medium for inter-tool information exchange (data and process integration). The data model consists of approximately 200 entities and 2500 relationships, derived from review of the MIL-STD-2167A. A Schema Definition Language (SDL) is defined which provides for project-specific modifications and integration of new tools with unique data requirements. For performance reasons, the SLCSE entity relationship data model is constructed on top of commercial relational databases.

Most COTS and government-furnished software tools which have been integrated into the SLCSE environment do not make extensive use of the SLCSE entity relationship database. Tools developed as part of the SLCSE project are primarily responsible for populating and utilizing the SLCSE database. These SLCSE-developed tools include document generators, requirements tools, design tools (text-based), problem-tracking tools, static analysis tools, and impact analysis tools.

The original SLCSE prototype operates in a VAX/VMS character-oriented terminal (VT100) environment. It defines a user interface format which specifies screen and menu formats, as well as navigation techniques. Tools are provided to simplify the integration of external (COTS or government-furnished) tools with the SLCSE environment.

SLCSE is currently being upgraded to reflect more current technology, including an X-Window user interface for bit-mapped displays.

4.3 VADS APSE

The VADS APSE environment effort is a commercial effort aimed at integrating selected COTS tools into the Atherton Software Backplane. The Atherton Software Backplane is an environment framework based on an object-oriented database supporting a single inheritance model. The Atherton Software Backplane provides capabilities for the creation of objects in the database, the definition of properties that distinguish between objects, messages to which objects can respond, and methods which define the response to a message. Relationships between objects are stored as a property of one (or more) of the objects. Messages and methods may be used to create “prologues” and “epilogues” which are sequences of activities to be carried out before and after an operation on an object.

In the VADS APSE integration, the objects defined in the Atherton Software Backplane database for a tool represent (at least in the limited integrations completed so far) the file system objects produced by the tool. The objects maintained in the Atherton Software Backplane for the Verdix compiler might be the Ada source code, the object files produced for each compilation unit, and the executable image generated by the linking phase.

The movement of objects (files) from the individual tools of VADS to the Atherton Software Backplane database has been automated. Rather than relying on the capabilities of individual tools, the Atherton Software Backplane provides multiuser support and version control.

In addition, strategic third-party tools, such as CADRE Teamwork and Frame Technology FrameMaker, are integrated. Future plans include integrating additional software development tools into the VADS APSE environment.

4.4 SoftBench

Hewlett Packard SoftBench provides control integration capabilities which are designed to complement data integration. These control integration capabilities provide mechanisms for tools to become active agents responding to service requests generated by other tools within the environment. This paradigm allows for a separation between agents (tools) that use a service, and those that provide a service. A similar paradigm is used in the X-Windows client/server relationship, and could be applied to other services used by tools, such as version control.

The Hewlett Packard approach to control integration focuses on the use of mechanisms called agents or triggers into which process information can be encoded for process integration. Agents or triggers are similar in concept to prologues and epilogues in the Atherton Software Backplane. While the Atherton framework concentrates on the data integration provided by the object base, Hewlett Packard aims to develop the use of agents and triggers to provide greater levels of control integration. Hewlett Packard has developed the Broadcast Message Server (BMS) to facilitate the intertool communication necessary for control integration. SoftBench also provides the Encapsulator to facilitate the incorporation of external tools into a SoftBench environment.

The decision to concentrate on control integration capabilities was reached primarily due to perceived difficulties in providing for true data integration. Hewlett Packard believes it has chosen an integration mechanism which will provide significant benefits regardless of which data integration gains market acceptance. In addition, Hewlett Packard is in the process of reimplementing Softbench using services provided by the European Portable Common Tool Environment (PCTE) to form an environment framework supporting tool federations.

Hewlett Packard is currently involved in cooperative efforts with tool vendors to build an integrated environment on top of SoftBench. Of particular interest is the development of a “virtual” versioning interface which has been used to integrate the various configuration management tools.

5 Observations on Tools and Environments

Sections 5.1 through 5.14 provide a summary of observations and trends noted based on surveys of environment builders and contact with tool vendors at workshops. The observations reported reflect the position of no particular vendor, but rather represent our perceptions of the current state and trends in tools and tool integration.

5.1 Maturing Tools

The early BASE environment efforts were particularly disadvantaged due to the limited availability of tools, and the limited functionality and relative instability of those that were available. The first available CASE tools around which the BASE environment was composed were characterized by sophisticated graphical drawing capabilities with little or no page layout capabilities, poor project management and software metrics support, limited code generation, and poor methodological guidance.

BASE developers had little choice but to incorporate specific tools because they were the only tools available on various Boeing development platforms. After suffering through frequent changes in programmatic interfaces in these tools, BASE developers began to ignore such volatile interfaces and chose to integrate early tools through a “least common denominator” approach of direct access to ascii data files.

More recent integration efforts (including current BASE efforts) have benefited from greater availability, increased functionality, and improved stability of tools. It is common for an environment builder to plan integration efforts for more than one tool of a particular class, thereby increasing environment flexibility and offering greater choice to the user. Through the natural maturation of tools and the computing environment, many tools now appear to have stable and robust programmatic interfaces and data formats, and offer adequate performance for small to moderately-sized projects.

The maturing of stand-alone tools has been accompanied by a number of changes to the scope and function of these tools. According to a number of the environment builders surveyed, tools which compete directly in a shared marketplace, such as IDE Software through Pictures and Cadre Teamwork have become increasingly similar by incorporating similar methods and technologies. For example, a review of Software through Pictures and Teamwork product literature [9, 10, 11, and 12] indicates that both products claim support for structured methods, real-time software development, MIL-STD-2167A style documentation, requirements traceability, entity relationship modeling, object-oriented design methodologies, Ada, and code generation.

The increasing similarity of high-level tool functions is not unexpected among tools in direct competition. Hewlett Packard engineers have also noted that the low level data objects and services offered by competing tools have become increasingly similar. As a result, a number of the surveyed environment builders felt that data interchange between tools of the same class is becoming increasingly possible.

Logically, the end result of providing tools of the same class with similar services is not to exchange data, but to make the individual tool plug compatible with similar tools in a SEE. The similarity of services within most classes of CASE tools falls well short of what is needed for plug compatibility. Plug compatibility will require additional agreements on what tools and services are to be provided. Unfortunately, no such service model exists, except for those tools falling into one of the service domains of the European Computer Manufacturers Association (ECMA) reference model for software environments [13]. Hewlett Packard is currently making use of the similarity in services offered by configuration management tools to develop a virtual versioning interface within the SoftBench environment.

The increasing maturity of tools and tool users also means that it is no longer adequate for a tool to be well engineered, stable, and provide similar services to other tools of its class. Tool vendors that wish to remain competitive must claim an “open” architecture to allow easy access to tool data and services. While Software through Pictures advertises “Visual Connections”, Teamwork advertises “Access” and Common Data Interchange Format (CDIF) compatibility.

Unfortunately, vendors interpret “openness” differently. An open architecture has been claimed based on providing modifiable data schemas, data access interfaces, externally visible data formats, programmatic interfaces, and customizable user interfaces, as well as through support for interchange formats such as CDIF. Such divergence in mechanisms for access of tools and tool data can only increase the complexity of integration efforts, and reduce the degree of integration possible.

5.2 Egocentric Relationships

Part of the legacy of the historical independence of stand-alone tools from environment technology is the egocentrism of the individual tools currently being integrated into tool coalitions. In spite of tool vendor efforts aimed at establishing such coalitions, each tool tends to view the tool universe as orbiting around itself. Due to marketing and performance realities, few tool vendors in coalitions will be willing to relinquish independence from other tools. This continuing tool egocentrism is characterized by the following:

- **Each COTS graphical tool in an integrated environment maintains its own user interface**. There is little or no attempt to generate a completely unified user interface. All environment builders noted that tool vendors are increasingly choosing to use the same mechanism (X Window) and look and feel guidelines (Motif). Vendors, however, have not agreed on conventions regarding operations available and menu format.

- **When viewed as tools, environments maintain a similar egocentric approach to the user interface**. All studied environments provide a common user interface format which allows certain tools (primarily batch or in-house tools) to be integrated at the user interface level, even when the environment was based clearly around a specific tool (such as VADS APSE).
- **Tools, even when claiming to be tightly integrated, maintain independent control of their own data and processing**. Environments tend to maintain separate, independent control of their own data and processing.
- **Each tool views itself as a "parent", while other tools are viewed as "children."** Invocations of tools occur primarily from parent to child.
- **Tool makers hesitate to change the approaches used and interfaces available within the tool at the behest of over tool vendors.** The attitude can be described as indifference.
- **Where tool functions overlap, the common approach is for each tool to continue to provide its own, often contradictory services.**

Environment builders surveyed maintain that the egocentric view of the software development environment maintained by many (if not most) tools is a major hurdle to more advanced levels of tool integration. The egocentrism of tools appears to be particularly troublesome for environments built on integration frameworks such as Atherton Software Backplane or Hewlett Packard Softbench. Softbench engineers suggest that in order to minimize tool egocentrism, tools should provide services such as modifiable user interfaces with extensible menus, mechanisms to specify where tool focus should be directed when the tool is invoked by another tool, interfaces for batch as well as interactive processing, and greater separation of services such as those specified in the ECMA model from individual tools. It is unclear whether tool vendors will surrender their egocentric positions, but the philosophy of federated tool environments directly challenges many of those positions.

5.3 Conservative Integration of Tools into Coalitions

Many of the stand-alone tool vendors who attended workshops indicated that they are beginning to offer coalition-style integration with complementary tools. In fact, this trend appears to be becoming almost universal among CASE tools. For example two of the largest CASE analysis and design tool vendors, Interactive Development Environments (IDE) and Cadre Technologies have begun marketing integrated environments. IDE provides a development environment composed of Software through Pictures, Sabre C, and Interleaf or FrameMaker. Cadre Technologies has announced integration agreements with Saber C and Pansophic, as well as offering access to the Interleaf and FrameMaker documentation tools.

While some of the current agreements between vendors to create tool coalitions are primarily marketing strategies, they do suggest that tool vendors now have increased confidence in the quality and stability of their products and the products of others. In fact, tool vendors appear to have solved many of the technical problems involved in building stand-alone tools and they indicate that market pressures are now forcing them to address the issues of long-lived, large-scale software engineering that inspired the early attempts to develop IPSEs. The result appears to be the distinct migration toward tool coalitions.

Tool vendors who attended the CASE Adoption and Tool and Configuration Management Workshops indicated that many of the current tool coalition integration efforts have taken relatively cautious approaches taken toward integration partners and modifications to tools. The types of tools and specific tools chosen for integration into coalition environments appear to be determined primarily by tool market share, and the competitive advantage offered by coalition-style integration. Thus, integration is becoming more common among tools with high market share in their respective areas, while tools with lesser market share are finding it harder to identify strategic partners for integration.

As a result, many of the tool coalition efforts focus on providing a programming environment rather than a full life-cycle product. Current coalition efforts often include analysis and design, documentation, and code generation tools. Tools which command more limited market niches, such as testing and code metrics tools, are often excluded from tool coalitions. In addition, tools useful in management of a project, but which require full life-cycle information, such as planning, tracking, and cost estimation tools, are not integrated.

The conservative approach is also evident in the type of Information that is extracted from one tool for use by another tool. Shared information is determined primarily by what is already available in tool interfaces, rather than by what would provide the best degree of integration. Changes to individual tools to support integration are either minimized, or performed only in support of high-leverage partners. Few vendors are willing to open up tool architecture enough to allow integration of other vendors with smaller market share.

Tool coalition integration efforts also make little attempt to identify and eliminate redundant services offered by multiple integration partners, or to provide a service framework to simplify the integration of additional tools. Thus, tools within tool coalitions may have conflicting development models and offer incomplete or conflicting configuration management, version control, and security services. Since individual tool vendors must compete as both stand-alone tools and members of a coalition, they are often unwilling to make risky concessions to consolidate services. Only in isolated cases where market consensus has developed, such as with windowing systems, do vendors migrate to a common service, and in those cases many tools within coalitions maintain an independent look and feel.

Due to the lack of standards and emphasis on “no change” policies, it can be expected that tool coalition integration efforts will provide an immediate, but only a partial solution for tool integration. Although CASE tool vendors are rapidly expanding into areas where tool support is now lacking, it is unlikely that a unified life-cycle product will result in the near future. The experience of environment builders suggests that such tool coalition integration efforts may be quickly limited by a lack of shared services. In addition, the lack of a shared design philosophy and shared services may also lead to inconsistency in the level of integration available in coalition toolsets.

5.4 Virtual Interfaces

The early experience of environment builders suggested that multiple operating platforms and substrate components were the rule rather than the exception for long-lived environments. For example, BASE migrated to three different hardware platforms, four different operating systems, two user interface systems, and two database systems during the life of the project.

Through experience, the BASE developers learned that virtual interfaces to substrate components could substantially aid in migration to different operating environments. The BASE developers designed and implemented a virtual operating system interface, which proved useful in subsequent migrations. They also made extensive use of the existing SQL standard to provide a virtual interface to the underlying database system. BASE developers noted that even with the SQL standard, the non-comprehensive nature of the standard made migration to a new database system difficult. However, the developers felt that without the standard, migration would have been far more difficult.

Other environment builders have experienced similar problems with migration. SLCSE engineers are now addressing a change in hardware (from a VAX to a Unix Workstation), operating system (VMS to UNIX) and user interface (character oriented to X-Window based). Fortunately, there is increasing agreement on a number of virtual interfaces for environment components. Environment builders surveyed either currently operate in a UNIX environment (Berkeley or System V variant), or have discussed plans to migrate to such an environment. With consensus building around POSIX compliance for UNIX and other systems (including VMS), a clear virtual interface between applications and the underlying operating system is possible. It is expected that such a virtual operating system interface will simplify porting of tools and environments from system to system. While increasing numbers of vendors are announcing POSIX compliance, it remains only a proposed standard, and is currently at the subcommittee level.

In providing a virtual versioning interface for configuration management systems, SoftBench engineers are taking advantage of developing consistencies in tool functionality to explore new areas for virtual interfaces. Such efforts, however, require significant cooperation on the part of vendors as well as consistency in basic services.

5.5 Toward "Small Scale" Data Interchange Formats

With almost all environment and tool integration efforts to date, a primary means of data integration has been interfaces to access tool data. While providing access to stored data, such interfaces normally convey little semantic content. They are also extremely unstable during early years of CASE tool development. Due to the instability of these early data interfaces, BASE environment builders chose to use ascii formats. As a result, they suffered from complex tool interfaces with poor efficiency of data transfer. Later, Boeing developed the Boeing Interchange Format (BIF) and used it to provide a common intermediate format for tool data.

The specialization of BIF to the Boeing environment and process allowed for the encoding of a limited degree of semantic information.

Currently, the CASE tool industry is focused on large-scale data interchange standards such as the CASE Data Interchange Format (CDIF). However, since they continue to address only the interchange of syntactic information, acceptance of such standards remains limited. Without a parallel transfer of semantic information, the benefits of CDIF are not clear to vendors. Other lower-level, smaller-scale, and domain-specific data interchange formats have been extremely successful in providing data interchange within a limited class of tools. One such successful standard is Postscript, which now is used almost universally by documentation systems. Another low-level standard generating interest is the Tagged Image File Format (TIFF), developed by Microsoft to simplify the transition of raster graphics between tools.

While evidence suggests that small-scale standards are not a panacea, some organizations currently building environment frameworks have expressed particular interest in them. They feel that acceptance of such standards can provide the mechanisms necessary for higher levels of integration, while circumventing some of the conflicting interests and limited benefits that make standardization of large-scale data interchange very difficult.

5.6 Consensus on User Interfaces

The “early” environments, including BASE and SLCSE, had a particularly hard time providing a common user interface across all tools in the environment due to the uneven maturation and acceptance of workstations with bit mapped displays and windowing systems. BASE and SLCSE were forced to operate in “mixed” environments with character-oriented and bit-mapped display technology evident in both hardware and tool design. Thus, the VMS character-oriented environment SLCSE was required to extract information from tools which ran on bitmapped displays.

The subsequent, almost universal adoption of bitmapped displays and the X Window system has simplified the problem of presentation integration for environment builders. Some environment and tool builders questioned indicated that the X Window System, along with the Motif toolkit, provided an adequate level of presentation integration.

Even with universal adoption of X Windows and Motif, presentation inconsistencies between tools are likely to persist. While Motif specifies the format of pulldown menus, it does not specify a standard for the contents of pull down menus or for the operations included in each menu. Thus, it is likely that individual tools will utilize different, and at worst, conflicting menus, operations, and control sequences. For Apple Macintosh-like consistency of user interface, additional levels of standardization are clearly necessary.

5.7 Isolation of Services from Tools

In spite of numerous differences in technology and implementation, BASE and SLCSE share a common view of the operating environment for tools. Both efforts reflect the more batch and character-oriented, less interconnected computing environments of the recent past. Both have suffered at times from limitations imposed by proprietary operating systems.

In many ways, the environment in which BASE and SLCSE were developed contrasts sharply with the current computing environment which is spawning a trend toward tool federations. The open systems approach toward computer systems and networking that snowballed during the late 1980s has led to general concurrence on an operating system (POSIX), mechanisms allowing applications using different protocols to communicate (XTI), and windowing environments (X Window System).

Perhaps most significant about the development of such open standards is the model that their adoption can provide to the tool community. Prior to the acceptance of X Windows, the provision of windowing services was often deeply embedded within, and unique to, each individual tool. The X Window System removes the responsibilities of windowing from the tool, and in doing so provides a classic example of the separation of the producer of a service from a consumer of a service. The client-server model provided by the X Window System allows the application to be executed on a device distinct from the device providing the windowing environment, leading to a form of application portability whereby a single application can be accessed from different platforms and architectures.

The Atherton Software Backplane and Hewlett Packard SoftBench are both rooted in the client-server approach to computing environments which has developed with open systems architectures. Both frameworks attempt to remove integration services from the domain of individual tools and instead provide such services as part of the framework. Atherton differs from SoftBench primarily in the service orientation (data management vs. tool control), and in the degree to which the client-server philosophy is adopted. Atherton attempts to provide both data management and control integration in a single framework revolving around an object-oriented database.

Unlike Atherton, Hewlett Packard chooses to isolate data management and control integration facilities as individual services, providing for independent development and replacement of the underlying database as technology matures. Hewlett Packard is further extending the service concept by providing a virtual interface for version control. A conceptual model for many such separate service domains is provided by [13].

5.8 Better Control Mechanisms

Control integration in the early environment BASE was provided by simple, script-driven spawning of batch processes. These mechanisms were primarily used for the periodic collection of data from individual tool databases, and for generation of documents and reports. The tools from which data was collected played at best a passive role in the process and in many cases played no role at all. Tools did not notify other tools of their actions and were often ignorant of the presence of other tools.

Surveyed environment builders and tool builders at workshops indicated that they have begun to adopt mechanisms which provide for more direct coupling of tools in heterogenous computing environments. Two such mechanisms are remote procedure calls (RPC) and Active Links.

RPCs are essentially the combination of a call protocol and a common data format. They represent a mechanism of invoking tools services distributed across heterogenous networks. By hiding the details of network computing, applications can tap into services provided by other applications which were previously unavailable. Service calls across applications appear similar to calling a local function. RPC mechanisms potentially allow the splitting of applications into smaller independent services.

Unfortunately, tool and environment vendors indicated that individual tool support for RPCs varies widely. Not only do tools differ in basic support for RPC (many tools offer no support), but no conventions have been established for the type of processing which should be remotely available. Environment and tool builders felt that such conventions will likely be specific to the type of tool, but may also be specific to the domain of the application program built with the environment (those domains which require MIL-STD-2167A compliance may enforce different conventions than those that do not).

Active Links provides a mechanism by which a tool can exert control over another tool to create the illusion of common data locality. They were originally conceived as a way of guaranteeing that the version of a picture or other insert in a document was the latest version available. Once an active link is established between an original document and the external source, any changes to the external source are reflected automatically in the original document.

Competing active link technologies are currently available in the FrameMaker and Interleaf documentation systems. FrameMaker's Live Links allows straightforward creation of links between tools supporting the Frame Technology format. Interleaf's Active Document format requires no special support from external tools, but is more complex to implement, since embedded programs are placed within the Interleaf document. With these programs, Interleaf documents can modify and format input data, protect themselves from unauthorized access, provide user-dependent views, and distribute themselves to reviewers.

The first system-generalized application of Active Link technology may not appear in Unix environments at all, but rather on personal computers. Apple now offers capabilities that link many Mac applications. In addition, Microsoft is developing a technology which will eventually provide transparent Active Links between IBM and Apple products.

5.9 Support for Software Process

Early environment integration attempts, represented by BASE, provided only primitive support for the software process. BASE encoded those portions of the Boeing BSWS-1000 process model primarily concerned with document structure into a set of standardized interfaces for tool invocation and into special integrating tools for producing documents and reports. The environment user interface rigidly enforced the original BASE process model.

Boeing's experience with this BASE encoding of the process model has not been entirely favorable. Because of the limited scope of the integrated environment (primarily oriented toward documentation artifacts), large parts of the development process remained outside of the BASE realm, thereby reducing the level of automation possible. In addition, orientation toward project deliverable in the original BASE models led to significant data redundancies. BASE developers also found that, due to individual preferences, hardware and software environments and special requirements of a funding agency, individual projects often required unique process and document support. Due to user dissatisfaction, Boeing later remedied the problems of data redundancy and a too-rigid process model by providing a stronger centralized database mechanism and removing the rigid encoding of the process model from the BASE user interface.

SLCSE, like BASE, has chosen to encode process information in the user interface and database. SLCSE, however, uses a more sophisticated, information-driven data model and provides a greater degree of flexibility in process support. The SLCSE database contains hundreds of object types, and thousands of relationship types derived from MIL-STD-2167A. An information-driven approach, which models low-level entities specified in MIL-STD-2167A rather than specific documents, was chosen to avoid data redundancy.

In SLCSE, multiple techniques are available to modify the encoded process model. The Winnie and Moo tools can be used to aid in the generation and modification of (character-oriented) user interface menus. In addition, approximately 17 roles derived from the Software Technology for Adaptable Reliable Systems (STARS) Operational Concept can be chosen to provide unique access control characteristics. However, most significant is the separation of the environment database into two separate databases for environment framework support and project support.

The environment framework database is designed to be relatively stable, and contain information about nodes, users, and tools common to all projects. The project database is designed to contain information unique to a project, and is expected to be heavily tailored. The two databases are both logically and physically separate for conceptual consistency and performance.

Unfortunately, since SLCSE has yet to be used in full-scale development, it is unclear how effective and well received the SLCSE process model will be. However, developers experienced with the SLCSE database have noted that due to unexpected complexities in modifying existing database schemas, a very small environment schema should be developed, and a new project schema should be built for each project.

VADS APSE, utilizing the Atherton Software Backplane, also provides process integration primarily through the user interface and database schema. Unlike SLCSE, Atherton provides a relatively well-developed mechanism to control the actions of individual tools through the environment object database. Relationships within the object base are used to encode the process model by specifying processing that should occur before and after an event. The Verdix designers have rejected the embedding of a specific process model such as MIL-STD-2167A.

They intend to provide such specific process models as a value-added service built on top of the core VADS APSE toolset.

HP SoftBench focuses on the control aspects of process integration. In addition to providing user interface support for user-defined processes, it provides a mechanism to notify (and initiate, if necessary) other tools in the environment of user-specified events. It is unique among the environments studied in that does not attempt to provide process integration through a data model. SoftBench developers chose to ignore data encoding of a process model due to the perception that environment database technology has not matured, and universally accepted data models have not been developed. Commitment to a specific data integration technology would be premature for these reasons.

5.10 Better Framework Support

Discussions with environment builders and workshop attendees indicate that tool integration frameworks have become viable due to the maturation of individual tools and the development of new database technologies. These new technologies, along with the booming tool marketplace, have encouraged large system integrators such as the Hewlett Packard, IBM, Digital Equipment, and Texas Instruments to enter the integrated tool marketplace. Other tool vendors, such as Verdix, have begun to integrate their tools and the tools of others with existing commercial framework services. The presence of such large vendors may indicate that a market shakeout has begun. In fact, engineers for the commercially-funded efforts surveyed (Hewlett Packard SoftBench and Verdix VADS APSE) expect the tools market to coalesce around a relatively few framework technologies.

Maturing database technology, as demonstrated in the SLCSE, VADS APSE, and also in PCTE environment efforts, has made support for increasingly complex data models feasible. The entity relationship data model supported by PCTE, along with the Object Oriented (OO) model supported by Atherton, are providing new, but unproven, possibilities for environment database management.

These advanced database technologies share a common characteristic of associating data with relationships and operations. In doing so, advanced databases provide a mechanism to establish relationships between tools. The database can become an intermediary in the environment, and lessen the need for environment knowledge on the part of individual tools. Semantic interpretation of information must remain in the domain of individual tools, however.

Both VADS APSE (Atherton) and SLCSE use advanced database interfaces. While VADS APSE uses the Atherton database to encode relationships between the coarse-grained artifacts of tools, SLCSE extracts information from individual tool databases, and encodes relationships between finer-grained objects. VADS APSE also attempts to make extensive use of the controlling capabilities of the database. The Atherton database underlying VADS APSE is used to associate controlling actions with operations on data objects in the object base. SLCSE uses an entity relationship database interface, built on top of a commercial relation database, primarily to associate data objects for traceability, impact analysis, and document generation.

Unfortunately, such database technology is only beginning to be used actual development projects. Implementing a data model based on MIL-STD-2167A, SLCSE has generated great interest, but is now only in a testing phase. The VADS APSE effort represents one of the first commercial attempts to use the Atherton Software Backplane as an integrating framework for a large-scale development environment. PCTE has been primarily used for small-scale research-oriented projects in Europe, and is only now generating interest among U.S. based environment developers, such as Hewlett Packard.

Hewlett Packard's SoftBench product represents a different approach to providing an integration framework, in which the forms of integration primarily concerned with tool control are provided as services separate from the environment database. The separation of data integration and control integration make SoftBench potentially compatible with a number of underlying databases. The decision to separate data and control services from the underlying database technology is apparently motivated by concern about the immaturity of environment database technology.

As with the Atherton Software Backplane, sequences of control actions can be specified in SoftBench, and associated with other events. In the SoftBench scheme, however, tools play a more active role in control integration. The SoftBench approach decidedly moves toward a revised schema in which tools are active agents in the integration environment, and as such are expected not only to send information about actions, but also to actively listen for information and requests from other tools.

5.11 Improved Data Flow Mechanisms

The earliest environments, including BASE, rely on the manual movement of data into the centralized database. In BASE, movement of data into the centralized database is accomplished by periodic execution of a batch process which extracts data from tool databases and stores it for subsequent processing. Movement of data from individual databases to the centralized database primarily occurs when the project is attempting to deliver a milestone document, or when specific information about the project is required. Consistency of data in the central database was maintained by the enforcement of policies for data updates. Such policies relied on manually enforced processes, supported by processes encoded in the BASE user interface. The central database provided little support for maintaining data consistency.

Migration of data to the SLCSE environment continues to be primarily accomplished through manual check-in of data. Like BASE, there is little environment support to guarantee that all data in the central database reflects the most current state of data in individual tools. Also like BASE, the database is not intended to be used for interactive development, but rather to reflect snapshots of a system at specific points in time. Thus in BASE as in SLCSE, individual tools continue to be the primary medium for designing and building a system in which data is collected into the central database to support special integrating tools such as document generators and impact analysis tools. However, SLCSE uses more advanced database technology to provide for better consistency checking in central database data.

An advance on this simple data migration mechanism used by BASE is offered by the VADS APSE environment. Rather than requiring manual migration of data from individual tools to the central database, VADS APSE automates data movement. When data is migrated to the central database from individual tools, other related data is marked for potential inconsistency, depending on relationships established between the objects. The automated mechanism for data migration potentially can make integration more seamless, and the presence of the central database more transparent.

5.12 Toward Finer Granularity of Data

The products or objects produced by individual tools that are directly available for transport to other tools have changed little over time. Both the earliest environment efforts, represented by BASE, and the latest efforts, represented by VADS APSE and SoftBench, complain that the available artifacts are primarily course-grained objects that reflect convenient units of information for representation in the tool database. In many cases, the units of information available are directly reflected in the physical organization of the tool database, e.g., one bubble in a dataflow diagram is stored per data file. Unfortunately, such course-grained data may be incompatible with the intentions of integrators, who often wish to access data at a finer granularity.

Faced with the problem of integrating coarse-grained data from individual tools into a unified database useful for the processing of integration tools, environment builders appear to have employed three approaches:

- Extract whole data objects from individual tool databases for incorporation into the central database.
- Decompose course-grained objects into finer-grained objects for incorporation into the central database.
- Incorporate complete data objects by reference in the central database.

SLCSE and BASE integrators extract and decompose data objects to support the data model defined in the central database. This approach has been found to be limiting in two primary ways: first, many critical aspects of tool data are maintained internally as conventions, or in other formats that proved difficult to interpret, and second, each individual extraction required intensive effort to implement and maintain. Often, when data is extracted, information is lost. Even in those cases where individual tools are designed to ease data extraction, contextual data is hard to maintain.

VADS APSE developers have chosen the approach of directly incorporating tool data objects by reference into the central database. While this approach eliminates the need for individual processing to support extraction, it does little to provide a finer level of granularity where it is needed. VADS APSE developers have indicated that where finer granularity is necessary, filters will be developed.

Some very recent advances have been made in providing fine- grained access to data items, particularly among tools that claim to be constructed with “open architectures” such as Interactive Development Environment’s Software Through Pictures, and the Procase Smartsystem. These tools have adopted strong programmatic interfaces that allow access to many of the data objects available within the tool. Procase, by way of a strong object orientation, claims to provide access to all objects and actions available within the tool. Unfortunately, fine grained access is only available from within the Procase Smartsystem tool set.

5.13 Support for Programming-in-the-Large

Historically, support for programming-in-the-large, including configuration management support, partitioning schemes for tool data, and project management support, has been a function of individual tools. Unfortunately, the support offered by individual tools appears to be inadequate. BASE developers maintain that they have on occasion exceeded the capabilities and capacities of every tool in the BASE environment.

The BASE environment itself provided little or no configuration, database, and project management support to facilitate the development of large systems. Relying almost exclusively on individual tool support, BASE developers found that tools must provide mechanisms to partition the tool database into multiple related databases on different devices. Unfortunately, few tools provided such mechanisms. Operations on the unified (non-partitioned) databases of tools proved impossible due to tool failure, or high cost of access time.

The consensus among tool vendors at SEI workshops generally supports the views of BASE environment builders. Tool vendors similarly feel that individual tool mechanisms to support programming-in-the-large are inadequate. They cite limitations within individual tools, variability between tools, and a lack of agreement on a development model as major causes of failure to scale up when tools are integrated.

More recent environment efforts often provide multilevel support for programming-in-the-large. In SLCSE, the central database is logically and physically separated into two distinct databases for conceptual consistency and improved performance. Configuration management support is also provided, but only for data that has been copied into the SLCSE database. A large majority of multiuser support, however, remains in the domain of the individual tools within the environment. Individual tools continue to maintain their own databases, and therefore the problems of scalability identified by BASE developers will continue to exist.

Integration frameworks such as Atherton Software Backplane are attempting to provide an additional layer of support for programming-in-the-large on top of the support offered by individual tool vendors. According to VADS APSE developers, integration of tool configuration management support with framework support has been straightforward. In many cases, Verdix chose to ignore the individual tool’s capabilities in favor of those provided by Atherton. It was not difficult, however, for Verdix developers to envision scenarios in which such integration may not be simple, particularly when individual tools maintain strong control over configurations and development models, or when they are tightly integrated with other tools providing configuration management.

Hewlett Packard's BMS system in Softbench is clearly designed to provide support for distributed tools, thus providing a useful mechanism for partitioning of tools and data. In keeping with the design philosophy of separation of services, however, Softbench provides no internal configuration management capability. Instead, Softbench offers integration with third-party tools providing such support such as Softool. It is not clear whether tool vendors will be willing to forego internal configuration management and partitioning mechanisms in favor of services provided directly and indirectly by Softbench.

5.14 Difficult Transition Issues

According to all surveyed environment builders and Case tool vendors, it is difficult to help organizations adopt CASE environment technology. Not only is the actual technology extremely costly, but it also requires a substantial modification to the manner in which the adopting organization does business. Some experts indicate that while the new technology is being adopted, productivity in the adopting organization may actually decline.

Among the surveyed organizations, only Boeing has substantial, long-term experience in transitioning the environment to users. The cost of the transition effort at Boeing was initially grossly underestimated, but now represents a large portion of the BASE budget. The BASE group is now involved not only in training, but also in the early planning stages of projects which will use the tool set.

Representatives for both the VADS APSE and HP SoftBench environment efforts expressed a good understanding of the costs of tool and environment adoption, and the importance of vendor participation in the adoption process. Both organizations offer or intend to offer considerable training and consulting help. It can be expected that such help may be particularly important to customers of the HP SoftBench product, which attempts to provide control integration mechanisms. VADS APSE customers may experience less upheaval in day-to-day activity, as the core toolset revolves around the well-understood compile-link-debug cycle. Verdix planners intend to function as integrators to provide more complete environment capabilities. Interest in the integrator role is also expressed by vendors of traditional analysis and design tools, such as IDE and Cadre Technologies.

6 Discussion

6.1 Current Tool Standardization Efforts

There are a large number of tool interconnection standards efforts currently in progress, ranging from extremely low-level standards to CASE-specific integration standards such as CDIF, PCTE, CASE Integration Standard (CIS), and A Tool Integration Standard (ATIS). They include government-backed, industry, and ad hoc standards efforts aimed at data management, tool portability, tool integration, and tool architecture. Among these efforts, no single standard is likely to supercede all other standards and independently guarantee future environment integration. Lessons from previous tool and environment integration efforts indicate that even old, low-level, and at first glance marginally relevant standards such as ASCII, Postscript, and SQL can prove important in tool and environment integration (a list of useful or potentially useful standards identified by the surveyed environment builders is provided in Appendix I). New standardization efforts are also likely to provide useful but not all encompassing pieces of the environment integration puzzle.

Unfortunately, the large number of standards efforts has confused tool vendors as well as tool users. As a result, vendors either show unwillingness to support any specific standards effort, or they choose to participate in several conflicting efforts in an attempt to “hedge their bets.” This choice reduces the risk that the vendor will adopt the “wrong” standard and also enhances the possibility that existing tool interfaces will approximate the (yet to be agreed upon) standard. While enhancing the position of the individual vendor, this posture actually lessens the probability of success for any specific standard effort by splitting the resources and commitment of the vendor.

With the large number of ongoing standards efforts, it will be extremely difficult for any group to participate in each one, or even to select which efforts among many others offer the greatest probability of success. As a result, de facto standards, based primarily on market clout, will likely emerge as major computer and tool vendors recognize the available market for integrated environments and become increasingly pessimistic about the chances for success of any specific standards effort. These de facto standards will likely incorporate characteristics of one or more of the proposed CASE standards.

The current and pending lack of homogeneity in environment integration standards will likely mean that interoperability between differing technologies and frameworks will be critical. Environment frameworks must be adaptable to a wide range of component technologies. For example, a framework must be adequately flexible to utilize relational, entity relational, and OO database technology, as well as multiple RPC and Active Link mechanisms.

If tool and environment users are to retain the gains made in the ongoing open systems movement, interoperability must also be insured between the surviving multiple, corporate-backed integrated environments and environment frameworks. At least one environment builder surveyed has already begun to consider integration between PCTE and Atherton-based standards.

6.2 Mechanisms vs. Process

The tool and environment vendors involved in various coalition- and federation-style integration efforts are clearly making significant progress in the development of mechanisms to support integration. This progress is readily apparent in coalition-style integrated environments, in new technologies for environment frameworks, and in such technologies as RPC and Active Links. While such technologies are necessary for integration, it is becoming increasingly apparent to environment builders that they are not sufficient for the production of a useful SEE.

What has been elusive so far in integration efforts is a consensus on how best to support process integration, and what processes are best to support. Surveyed environment builders reported many problems with process integration and few solutions. For example, from BASE and SLCSE efforts we can learn that the tightest level of process integration is available if the domain is limited. For BASE, a limited domain based on the Boeing methodology meant that more semantic information could be encoded in BIF, and subsequently made available for other tools. In addition, tight user interface enforcement of policy was possible. Unfortunately a second lesson learned from BASE is that such tight enforcement of policy, particularly in the user interface, is not well received.

Environment efforts which are providing advanced mechanisms for process integration, such as VADS APSE and SoftBench, are also experiencing the same problems determining an appropriate process to integrate. VADS APSE developers have chosen to ignore the more contentious areas of process and focus on the relatively well understood development cycle. Hewlett Packard engineers expressed concern about identifying a process or small set of processes that would be acceptable to most software developers.

One clear focus for organizations wishing to provide a future SEE should obviously be clear definition of a process or set of processes which form the core of engineering activities. In addition to identifying such processes, organizations could begin by restructuring in-house tools to support process-providing environments. Among tool features that can support such encoding of an organization’s software process are configurable menus, conventions for tool construction such that tools are organized around a number of service provision centers, and support for new process-encoding mechanisms like RPC and Active Links.

An organization can perhaps exert greatest influence on the direction of de facto standards efforts by influencing the direction of corporations currently involved in building environments, rather than by building unique environments. Our survey of environment builders suggests that support for software process is a foremost concern for both software organizations and for environment builders. This confluence of interest offers a unique opportunity to insure that the process concerns of the software industry are addressed in the products of environment builders.

6.3 The Cycle of Technology

The study of individual environment efforts has uncovered a phenomenon in which advancements in one of the various forms of mechanistic integration (data, control, and presentation) has led to a recognition of inadequacies, and subsequently new demands, on that same and other forms of mechanistic integration. Thus, when BASE developers working with tool vendors gained access to data within tools, it became apparent that the control and user interface mechanisms available were inadequate for a sufficiently flexible encoding of the Boeing process. In addition, access to the syntax of data pointed out the need for greater access to the semantics of data, as well as the need for new mechanisms to insure the consistency of data.

Likewise, as new data models, such as OO model, and new control mechanisms such as RPC and Active Links are becoming available, it is increasingly evident that the granularity of data available from tools is inadequate, and thus the programmatic interfaces of tools must be enhanced. However, increasing the granularity of data will increase the performance requirements on databases.

As new control mechanisms become increasingly available, tool vendors will be pressured to provide access to processing deeply embedded within the architecture of the tool. For example, an integrator may wish to gain control from a tool between the point where the user enters the command within the tool, and the corresponding action is carried out. A transfer of control at this point would allow the integrator to check for consistency across related databases prior to the tool performing the action. Unfortunately, few tools are currently structured to allow for the intercepting of requests between the user interface and the tool functionality. Such separation may be necessary for fine-grained control of the interactions between tools.

The increasing cycle of demands on integration mechanisms and individual tools may drive tool vendors toward support for a federated approach. It is unlikely that individual vendors can keep up with these increasing demands without reaching some consensus on services to be provided by a framework. Framework support will insulate individual tools to some degree from the implementation of underlying data, control, and presentation mechanisms, as well as from the operating environment. The consensus that has developed around the X Window System provides an early example of such framework support.

6.4 Tool Coalitions or Tool Federations?

Analysis of the integration efforts carried out by environment builders and tool vendors indicates that they are trying to solve multiple problems, including the integration of tools, the development of a supporting framework, identification of standards to provide for tool replacement, and the provision of portability for the integrated environment. The primary differences between IPSE approaches, tool coalition approaches, and tool federation approaches are greatly determined by where emphasis is placed in solving these problems.

The IPSE approach focuses on the provision of services by the environment. Tools were perceived as subservient to, and entirely dependent on the environment. By this separation of the tools from the underlying substrate, portability could be provided through porting of the IPSE services. With tool coalition efforts, the emphasis is placed on providing a specific instance of tool-to-tool integration. Sharing of services occurs based on the immediate needs of the tools in the coalition, and not on any underlying philosophy of environment integration. Plug compatibility of tools within the coalition environment is not addressed; in fact, it may even be discouraged. Portability depends on gathering a consensus around a common, low-level environment, including specific architectures, operating systems, and windowing systems. Tool Federation efforts focus on both tool-to-tool and tool-to-integration framework issues. They, like tool coalitions, are benefiting from the consensus on low-level interfaces for operating systems, hardware architectures, and windowing systems. They are also benefitting from the work done by individual tool vendors and tool coalition partners in providing tool services to support integration. Tool federation vendors intend, however, to usurp some of the responsibility for such support from individual tools and offer “value added” capabilities on top of other services. Ultimately, tool federations promise to offer greater flexibility for tool users, as well as support for plug compatibility. Unfortunately, tool federations are just beginning to develop.

In the short term, tool coalitions will likely provide a pragmatic link between the immediate demand for tight integration and the desire for a more generalized and adaptable Tool Federation solution [5]. Such coalitions, however, will likely be limited to supporting only well understood processes within the life cycle, such as the generation of standard documentation, and the analysis/design/implementation cycle. That part of the process supported by tool coalitions will be tightly embedded within the tools in the coalition, and therefore relatively inflexible. It also appears likely that many critical areas, such as project management and testing, will go without even this inflexible support.

7 Conclusion

Perhaps the primary lesson to be learned from the study of environment efforts is that no single solution to any existing problem, and no single interface, is likely to “solve” the tool integration problem. The different environment efforts have emphasized different mechanisms over time, yet the primary problems involved in providing flexible support for software engineering remain. A potential reason for the difficulty in defining a universal solution may be found in the general tension that is endemic to tool and environment integration. Builders of tools and environments must attempt to find a balance between a number of conflicting demands, including:

- Process enforcement vs. flexibility
- Tool integration vs. replaceability
- Process, data, and presentation integration, and,
- New vs. proven technology.

The failure of the software engineering community to progress toward acceptable levels of integration can be contrasted with the apparent success of the CAD community in defining appropriate standards for tool integration. While the software engineering community debates languages, methodologies, standards, and even life-cycle models, the CAD community’s success appears to be due in large part to a general agreement among CAD professionals on a process to be supported. The lack of an accepted process for software engineering has forced environment builders to attempt to choose between enforcing a process that lacks consensus and providing little direct process support.

The resulting environment implementations can provide useful lessons. The early BASE experience with strict encoding of a process model indicates that such encoding proves too restrictive to be adapted in multiple situations. It is unclear whether the degree of process flexibility supported by SLCSE is adequate. The SoftBench approach of providing control primitives appears promising, but tools may not be sophisticated enough to take advantage of the services offered.

The requirements for tight integration between tools and for replaceability of individual tools in the environment appear to present a major design conflict for environment builders. Early experience with BASE indicated that, even with shared “standard” interfaces such as SQL, tools could not be easily replaced. Currently, the primary mechanisms used to get tighter integration between individual tools are vendor agreements to form tool coalitions. While this approach does lead to more tightly-coupled tools, it effectively makes each tool an irreplaceable component of the SEE. For tools to become both tightly integrated and replaceable, major agreements must be reached not only on the role of each tool and on (near) identical interfaces available within the tools, but agreements must be reached on the provision of framework services to those tools. Groups deciding on standards must in effect balance conflicting needs: to ensure a tightly integrated environment for the near- to medium-term future, and to be able to adapt such an environment to changing demands of the framework, changing technology, and changing tools.

In attempting to formulate a strategy for a SEE, an organization must also attempt to balance the control, data, and presentation technologies provided. This task is made essential by the fact that not all parts of a SEE integrate in the same manner. For example, while compiling and debugging can be integrated by use of shared data structures, traceability and design require integration through control mechanisms and data interchange. A successful SEE will be one that provides a balance of flexible mechanisms to support control, data, and presentation integration. Unfortunately, such balance between mechanisms is hard to achieve, as each new technology has the potential to modify the demands on other technologies, as RPC technology has modified the demands placed on programmatic interfaces.

In addition, environment builders must select between support for the newest technology available or for more proven technologies. For example, some very useful tools use older technology, including Domain Software Engineering Environment (DSEE) and SLCSE. At the same time, we are attempting to incorporate two new models of database support, entity relationship, and OO. It is unclear at this time which technology will predominate or whether a new and better technology will be discovered. For many such cases of unstable technology, the most successful strategy may be to maintain adequate flexibility to adapt to any potential solution.

In finding ways to balance the many conflicting demands of SEE planning, perhaps the primary goal must be to maintain flexibility. It can be expected that new tools, integration technologies, and processes will emerge to supplant current ones in the following years. A successful SEE likely will be one that can adapt to these new tools, technologies and processes.

Appendix A   Identified Integration Standards

Framework Standards

- Portable Common Tools Environment (PCTE)
- A Tool Integration Standard (ATIS)

Operating Systems Standard

- POSIX

User Interface Standards

- X Window System
- Motif
- Open Look

Database Query Language Standard

- SQL

Data Interchange Standards

- CASE Data Interchange Format (CDIF)
- ASCII

Documentation Standards

- Postscript

New Standards and Conventions Necessary

- Tool Composition
- Configurable Menus
- “Degraded Mode” Processing (Information Missing)
- Program Start-up with Specific Focus
- Inverses to all Actions
- Event Notification

References

1 Dowson, M., “ISTAR - An Integrated Project Support Environment,” in *Proceedings of the 2nd SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments*, pages 27-33, December 1986.

2 Thomas, Ian, “Tool Integration in the PACT Environment,” in *Proceedings of the 11th International Conference on Software Engineering*, pages 13-22, May 1989.

3 Archer, J.E. and Devlin, M.T., “Rational’s Experience Using Ada for Very Large Systems,” in *Proceedings of the First International Conference on Ada Programming Language Applications for the NASA Space Station*, NASA, June 1986.

4 Requirements for the Ada Programming Support Environment: Stoneman, Department of Defense, February 1980.

5 Wallnau, K., and Feiler, P.H., “Tool Integration and Environment Architectures,” SEI Technical Report CMU/SEI-91-TR-11, May 1991.

6 Wasserman, A., “Tool Integration in Software Engineering Environments,” in *Lecture Notes in Computer Science*, #467, Springer-Verlag, Fred Long, ed., ISBN 3-540-53452-0.

7 Strelick, T., “The Software Life Cycle Support Environment (SLCSE): A Computer Based Framework for Developing Software Systems,” in *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, Boston, Massachusetts, Nov 28-30, 1988.

8 Cagen, M.R., “The H.P. SoftBench Environment: An Architecture for a New Generation of Software Tools,” *Hewlett-Packard Journal*, 41, 3, June 1990, pages 36-47.

9 *Software through Pictures, Products and Services Overview*, Interactive Development Environments, 1988.

10 *Software through Pictures, Ada Development Environment Product Announcement*, Interactive Development Environments, 1989.

11 *Unified CASE*, Cadre Technologies, 1989.

12 *Product Overview*, Cadre Technologies, 1989.

13 Earl, A., “A Reference Model for Computer Assisted Software Engineering Environment Frameworks,” Proposed Technical Report, August 17, 1990, Version 4.0 ECMA/TC33/TGRM/90/011.

UNLIMITED, UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S CMU/SEI-91-TR-13	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-91-TR-13
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (City, State and ZIP Code) ESD/AVS Hanscom Air Force Base, MA 01731
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office 8b. OFFICE SYMBOL (if applicable) ESD/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO: 63756E PROJECT NO.: N/A TASK NO: N/A WORK UNIT NO.: N/A

11. TITLE (Include Security Classification)
Case Studies in Environment Integration

12. PERSONAL AUTHOR(S)
Ed Morris, Peter Feiler, Dennis Smith

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM TO	December 1991	36

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse of necessary and identify by block number)
FIELD	GROUP	SUB. GR.	CASE, environment integration, tool integration, environment standards

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Four environment builders and participants at two workshops were queried concerning the environment standards, implementations, and technology that prove useful in the integration of tools into software engineering environments. Specific information was gathered about the software and hardware environments in which tool integration occurred, the goals of integration, the tools integrated, mechanisms used, and the standards applied. Observations concerning the current state of tool and environment integration are provided, and trends in integration are identified.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution
22a. NAME OF RESPONSIBLE INDIVIDUAL John S. Herman, Capt, USAF	22b. TELEPHONE NUMBER (Include Area Code) (412) 268-7631 22c. OFFICE SYMBOL ESD/AVS (SEI)

DD FORM 1473, 83 APR EDITION of 1 JAN 73 IS OBSOLETE UNLIMITED, UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS

ABSTRACT —continued from page one, block 19