Computer Standards & Interfaces 15 (1993) 431–443
North-Holland

Issues in the definition of a project support environment reference model

A.W. Brown [a,*], D.J. Carney [a], P.H. Feiler [a], P.A. Oberndorf [b] and M.V. Zelkowitz [c]

[a] *Software Engineering Institute, Pittsburgh, PA 15213, USA*
[b] *Naval Air Warfare Center, Aicraft Division, Code 7031, P.O. Box 5152, Warminster, PA 18974-0591, USA*
[c] *Computer Science Dept., University of Maryland, College Park, MD, USA*

Abstract

In an effort to establish interface standards to help the US Navy more easily and effectively assemble software-intensive Project Support Environments (PSEs) from commercial sources, the Navy's Next Generation Computer Resources (NGCR) program has set up a Project Support Environment Standards Working Group (PSESWG). The foundation of this group's work is the development of a service-based reference model that will provide context for categorizing and relating existing standards and the identification of interface areas that may benefit from future standardization. This paper presents a report on the lessons learned in the definition of this reference model.

Keywords. Project Support Environment; service-based reference model; framework services; data interfaces

1. Introduction

In industry, government, and military circles the need for improved approaches to computer system support for large-scale software development and maintenance has long been recognized. As early as 1979, a US Government Accounting Office study of nine software projects showed that the cost and destiny of software was:

- $3.2M – paid for, but never delivered
- $2.0M – delivered, but not used
- $1.3M – abandoned or reworked
- $0.2M – used after changes
- $0.1M – used as delivered.

While there have been improvements over the past decade there are still many examples that confirm the magnitude of the problem with which we are currently faced.

One way to address this problem is by improving the support environment used by large software projects. This overall support environment (hardware, software, methods and techniques, people) can have a significant effect on the quality of the computer system developed, and the ease (or otherwise) with which it can be maintained. The software aspect of a support environment is the primary focus of this paper.

The term Project Support Environment (PSE) refers to a computer-based system used in developing, maintaining, and enhancing computer systems. PSEs are currently being studied and used by many organizations in government and industry. Many organizations are seeking ways more easily to develop PSEs that are specific to particular projects or individuals. The goal of most of these efforts is finding a strategy that permits a PSE to be constructed from commercial off-the-shelf (COTS) tools in a flexible and maintainable way. Unfortunately, while sound in concept, this approach suffers from the current instability and fragmentation of the COTS tool market, with the result that assembling a PSE from a collection of COTS tools is a very complex undertaking. Not only are there many different COTS tools to choose from, there are many tools offering similar functionality, new tools being announced by vendors on a frequent basis, and no established means to use multiple tools within a single PSE. While often talked about, the notion of 'plug-

* *Corresponding author.* Fax: 1 412 268 5758, email:awb@sei.cmu.edu

and-play' compatibility in COTS tools remains a long way from reality.

As a means to solve this dilemma, many working groups are investigating how standardized interfaces (actually sets of complementary interfaces) can provide the necessary mechanism for tool integration. As a first step, it is necessary to isolate the specific areas for which interface standards are needed. A suitable abstract model of an environment is a prerequisite in accomplishing this.

1.1. Background to the reference model

Next Generation Computer Resources (NGCR) is a US Navy program designed to establish industry-based interface standards in a number of areas important to mission-critical computer resources (MCCR). Recognizing the current state of the practice in the area of support environments, the Navy decided as part of the overall NGCR program to focus one of its teams on the area of PSEs [1]. It consists of participants from industry and academia, as well as a variety of government services and agencies. The PSE Standards Working Group (PSESWG) is selecting interface standards that will help the Navy in moving toward the goal of being able to assemble a PSE from COTS tools in a well-defined way.

The PSESWG was initiated in February 1991 with a charter to establish an industry-based set of environment interface standards. These standards, and the environments that conform to them, must be suitable for supporting engineering and management through the entire life-cycle of computer-based applications systems in the 1990s and beyond. Two related tasks were initiated as a starting point to achieve the PSESWG charter:

(1) The development of a PSE reference model. Due to the complexity and lack of agreed terminology and concepts in this area, it was decided to develop a model based on the characterization of the facilities expected of a populated PSE. These facilities include both the support services and the tools that provide capabilities to the end-user.

(2) A cataloging of available technology and standards. As there are many existing standards and products already in use, there was a need to begin the task of identifying and categorizing those that might be relevant to the PSESWG.

There is a close relationship between these two tasks in that the aim is to validate the reference model by applying knowledge of existing products and standards, and to use the categories and partitions of the reference model to help in structuring the available technology catalog.

The PSE reference model is the starting point for selecting interface standards. It will provide the context for classifying existing products and standards, establish a common terminology and set of concepts for PSESWG (and perhaps the broader PSE community), and identify where further standards efforts may need to be directed. In developing this reference model we examined a large number of existing PSE efforts. None of them individually had the breadth, nor had the approach necessary to achieve our aim of providing a vehicle for identifying interfaces as potential candidates for standardization in a PSE. Hence, our model synthesizes aspects of many of them, including the Software Technology for Adaptable, Reliable Systems (STARS) program, the National Institute of Standards and Technology (NIST) Integrated Software Engineering Environment (ISEE) working group, the European Computer Manufacturers Association (ECMA) Technical Committee 33 task group on the SEE reference model, the Ada Joint Program Office Evaluation and Validation Team, the Air Force Software Life Cycle Support Environment (SLCSE) project, Honeywell's Engineering Information System (EIS) program, TRW's Conceptual Environment Reference Model (CEARM) effort, and the standardization committees within IEEE and ANSI for POSIX and for CASE Tool Integration Models (CTIM). Many valuable aspects of these efforts have been considered in the course of our work.

The initial use of the PSE reference model will be as the basis for identifying interface areas where standards are likely to be of benefit in assembling a PSE from COTS tools. However, the reference model may also be seen as a definition of common terminology and concepts that will allow existing PSE products and standards to

[1] In addition to the PSESWG, other working groups are concentrating on network, backplane bus, operating systems, database, and graphics interface standards.

be described and compared. In this regard potential users of the reference model include tool builders, tool vendors, environment integrators, and the software engineering community as a whole.

The task of writing the reference model is [2] one of significant technical difficulty, given that the chosen job is to create a valid model for any PSE. However, the actuality of the task is that a combination of technical, political, and managerial skills is needed for such an exercise. Factors such as the inherent makeup of the NGCR working groups, the relationship between the PSESWG and other groups in the environments area, and the current ferment throughout the software standards community are all contributors to the difficulty of developing such a reference model.

1.2. Overview of this paper

The rest of this paper focuses on two distinct topics:

(1) A review of the main elements of the reference model.
(2) A discussion of some of the main issues that were addressed in defining and applying this reference model.

The first of these topics is addressed in sections two and three. Section 2 describes the basis of the reference model, together with a description of the key terms and concepts that provide a basis for this work. Section 3 describes the services defined in the model itself.

The second topic is addressed in Section 4, which consists of a detailed commentary on some of the main lessons learned in defining and applying this reference model.

The paper is concluded in Section 5 which summarizes the main issues raised and describes ongoing activities related both to the reference model and to other related research activities.

2. Description of the model

The reference model is a conceptual description of the functionality that may be provided by a project support environment [3]. This description is general and is bounded neither by a particular application domain nor by any specific life-cycle paradigm for a development project. This is in contrast to an actual implemented environment that is constructed of particular components (i.e. software and hardware) and that typically does reflect a chosen lifecycle paradigm, at least implicitly.

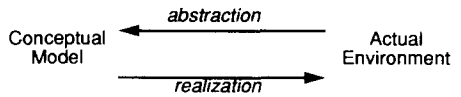


Fig. 1. Conceptual models vs. actual environments.

The distinction between conceptual and actual is of fundamental importance. The conceptual viewpoint that governs this reference model provides an abstract description of the functionality that may be found in a PSE. An actual viewpoint would describe a particular realization of the conceptual view in terms of a PSE architecture with specific tools and standards. There is a mutually reflective relationship between the conceptual and the actual views, i.e. between this PSE reference model and existing environments: one may either consider the model to be abstracted from many environments, or may regard a particular environment as a realization of the model.

Figure 1 illustrates this distinction. The left-pointing arrow illustrates the activity of studying several existing environments to derive information for the model. The right-pointing arrow indicates that a particular environment could be a realization of the model. One benefit of this approach is that it provides a common means of describing environments (e.g. 'In terms of their functionality, how is SLCSE [4] different from EAST [5]?'). Hence, the reference model does not directly represent an architectural approach to constructing a PSE – rather, it provides a common basis for examining the functionality of different PSEs. Additionally, such an analysis pro-

[2] This is an on going task, since although a public document now exists (Version 1.0), the reference model is still in development. The completion date of Version 2.0 is October 1993, with periodic revision thereafter.

[3] Although the term 'environment' has not yet been fully defined, the reader is presumed to have some familiarity with the term as commonly used.

[4] The Software Life Cycle Support Environment, a software development environment sponsored by the US Air Force.

[5] The Environment of Advanced Software Technology, a software development environment product developed by SFGL in France.

vides an ongoing validation of the model; it is a necessary attribute that the reference model provides an accurate reflection of technology that exists.

2.1. Key concepts and terms

There are several key concepts and terms used in the reference model. This section provides an overview of them and their interrelationships. These key terms are indicated below by italics. It should be noted that some of these concepts are not amenable to simple definition, either because the term is broadly applicable, forcing description rather than definition, or because the term currently has conflicting meanings in the environments community. It is hoped that this section of the paper may help resolve some of this confusion.

An *Environment* is a collection of software and hardware [6] components; there is typically some degree of compatibility that renders these components harmonious. One can describe an environment using the contrasting viewpoints of conceptual vs. actual; or in a slightly different way, one can describe an environment in terms of the way it supports human activities.

When described from the conceptual point of view, an environment's capabilities are referred to as *Services*, which are abstract descriptions of the work done. Some of these services are of direct interest to an *End-user* (e.g. editing) while others comprise an underlying infrastructure, or *Framework*, comprised of relatively fixed capabilities that support user interfaces, processes, and objects (e.g. access control, process resource management).

When described from the opposite, or actual view, i.e. when a realized environment is considered, the components that directly support an end-user are generally called *Tools* (e.g. graphical design packages). Although no single definition for 'tool' will suffice, that of the IEEE Glossary [7] is useful: a computer program used to help develop, test, analyze, or maintain another computer program or its documentation. As in the conceptual view, the components that comprise an actual infrastructure are referred to as the *Framework*. The same term, framework, is thus used in both a conceptual and an actual sense, and its precise meaning depends on the context.

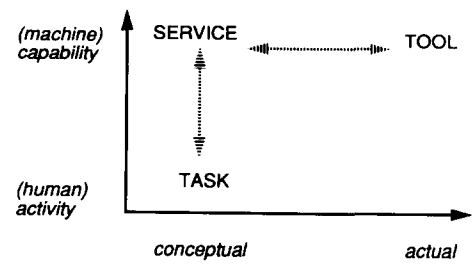


Fig. 2. Relationships of tools, tasks, and services.

Finally, when an Environment is considered from the vantage point of how it supports human activities, then either the environment will provide a *Service* to a human user, or a human user will perform some *Task* with the aid of the environment. For instance, one can speak of the *task* of testing software, or of using a software testing *service*. These different views of an environment result in subtle differences in the meanings of key terms. In particular, there is a slightly different meaning for service when it is contrasted with tool and when it is contrasted with task. In the first case, a tool is an actual realization of one or more conceptual services.

While there is no strict correlation between tool and service (because one tool may realize many services, or one service may be realized by many tools), there are relatively straightforward correlations between tools' functionalities and service descriptions. In the second case, a task and a service provide complementary views of the same activity. For instance, one might consider that the environment provides some capability (e.g. an environment's testing service); or one might consider that a human user performs some task using the environment (e.g. the human activity of testing). Whichever view one takes, both refer to the same notion, (e.g. a human using a piece of software to test the output of an engineering process).

In brief, services are abstract capabilities of the environment, tasks make use of and provide context for those capabilities, and tools are the actual executable software components that realize environment services. *Figure 2* illustrates the

[6] For the purposes of this document, the PSESWG concentrates on the software components of an environment.

[7] *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std 610.12-1990.

distinction between these concepts. *Service* can be contrasted with *Tool* along an axis of Conceptual vs. Actual, or it can be contrasted with *Task* along an axis of Capability vs. Activity.

The PSE reference model is a catalog of service descriptions spanning the functionality of a populated environment. The service descriptions are grouped by several different categories, including degrees of abstraction, granularity, or functionality. The highest-level division classifies services either as end-user or framework services. The former includes services that directly support the execution of a project (i.e. services that tend to be used by those who directly participate in the execution of a project such as engineers, managers, and secretaries). The latter services either pertain to users who facilitate, maintain, or improve the operation of the computer system itself (e.g. a human user performing such tasks as tool installation) or are used directly by other services in the environment. End-user services are further subdivided into Technical Engineering, Technical Management, Project Management, and Support services. The first three of these groups partition the execution of a project into engineering, management, and a middle category that partakes of both. The fourth group, Support services, is orthogonal to the other three, since it includes capabilities potentially used by all other users, such as word processing, mail, and publication.

Figure 3 illustrates the logical relationships between these service groups. Framework services form a central core with a potential relationship to all other services in the environment. Support services underlie the other end-user services. The remaining three groups, Technical Engineering, Technical Management, and Project Management, surround the Framework services and make use of the Support services. In addition, services from these three groups may have relationships with each other. It is not the intention that the boundaries of the parts of this drawing explicitly indicate interfaces, since this figure is drawn at the level of service *groups*, not of individual services. Thus, it must be stressed that while a drawing such as this attempts to suggest in a very general manner how the high-level service groups logically relate to each other, there is an express intention to avoid any sort of architectural implication. The reference model is a conceptual, not an actual, model, and no architectural choices are intended by this figure. To emphasize this point the same set of service groups, with the same set of potential relationships, could also be illustrated by *Fig. 4*.

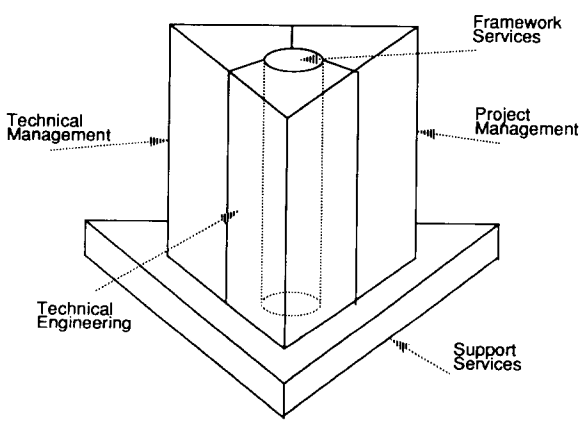


Fig. 3. A graphical depiction of the reference model service groups.

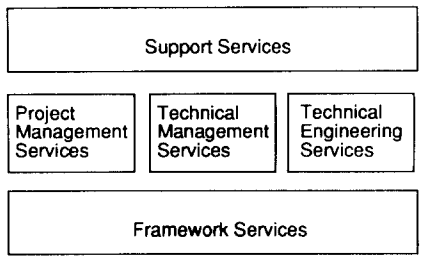


Fig. 4. An alternative view of service groupings.

The key point is that the figures are illustrative only and do not in any way connote layering of services, architectural decisions, or implementation choices for an actual environment.

3. Description of reference model services

The reference model distinguishes two groups of services: end-user services and framework services. In this section we briefly review the services that have been defined in each of these groups, and examine the distinction between host and target system support in a PSE. Full details of the reference model services can be found in the complete reference model document [5].

3.1. End-user services

Each of the end-user service categories (Technical Engineering, Technical Management, Pro-

ject Management, and Support services) is further subdivided by engineering domain, by user role, or life-cycle phase. Technical Engineering services focus on the technical aspects of project development. These services support activities related to the specification, design, implementation, and maintenance of systems. They are subdivided by specific engineering domains (e.g. Software Engineering). In addition to 'traditional' engineering domains, the reference model also considers life-cycle processes to be an area for which an engineering discipline is appropriate, and services related to that domain are included here as well. Within an engineering domain the processes used in the life cycle of a project define a series of tasks, each requiring services for its support. Thus, within the software engineering domain, tasks typically include designing and coding, which require services like compilation and testing. The following services are defined in the reference model:

System engineering services
- System requirements engineering
- System design and allocation
- System simulation and modeling
- System static analysis
- System testing
- System integration
- System re-engineering
- Host-target connection
- Target monitoring
- Traceability

Software engineering services
- Software requirements engineering
- Software design
- Software simulation and modeling
- Code verification
- Software generation
- Compilation
- Debugging
- Software testing
- Software static analysis
- Software build
- Software reverse engineering
- Software re-engineering
- Software traceability

Life-cycle process engineering services
- Process definition
- Process library
- Process exchange
- Process usage

Technical Management provides services that are closely related to engineering activities. However, these services pertain to activities that are often equally shared by engineers and managers; the operations of these services do not clearly fall into one or the other category, but fall into a middle category that partakes of both Technical Engineering and Project Management. These services provide a managerial complement to engineering activities in the areas of Configuration Management, Reuse, and Metrics.

The following Technical Management services are defined in the reference model:
- Configuration management
- Change management
- Reuse management
- Metrics

Project Management services are relevant to the overall success of the enterprise. These services support activities related to developing and executing a project, including such things as scheduling, planning, and tracking the project's overall progress. These activities span the lifetime of a project from inception through deployment and maintenance.

The reference model describes the following Project management services:
- Scheduling
- Estimating
- Risk analysis
- Tracking

Support services focus on tasks and activities common to all users of a PSE, regardless of the domain, role, or life-cycle phase in which the activity is taking place. Support services are needed by virtually all users of the computer system. They include services associated with processing, formatting, and disseminating human-readable data, including several common text and figure processing services, as well more specialized publishing, user communication, and presentation services. They also include administration services that provide support for use of the PSE itself.

The reference model describes the following Support Services:

Common support services
- Text processing
- Numeric processing
- Figure processing
- Audio and video processing

Calendar and reminder
Annotation
Publishing service
Presentation preparation service
User Communication services
Mail
Bulletin board
Conferencing
Administration services
Framework administration and configuration
Environment administration

3.2. Framework services

These services comprise the infrastructure of a PSE. They include those services that jointly provide support for applications, CASE tools, etc. and that are commonly referred to as 'the environment framework'. Since 1989, the National Institute of Standards and Technology (NIST) has sponsored a series of workshops developing a reference model for environment frameworks. The product of that group is a document published jointly by NIST and the European Computer Manufacturers' Association (ECMA), and is commonly known as the 'NIST/ECMA Frameworks Reference Model' [2]. This document contains detailed descriptions of fifty framework services. The PSESWG elected essentially to use the NIST document in its entirety, and the PSESWG reference model simply contains abstracts of the more extensive descriptions found in the NIST/ECMA document.

In addition to the NIST/ECMA set of framework services, the PSESWG has also chosen to include some other infrastructure services not present in the NIST/ECMA document. The PSESWG has abstracted several services from the work of the 'Draft Guide to the POSIX Open Systems Environment' sponsored by the IEEE, known as 'POSIX.0,' as a source for these [4].

In both cases, the PSESWG reference model has abstracted the service descriptions. A reader of the PSESWG reference model is urged to consult these other two documents for a full description of the infrastructure services. It should also be noted that, at the infrastructure level, some services are actually groups of services which in turn contain lower-level services.

The reference model defines the following framework services:
Operating system
Object management
Process management
Policy enforcement
User interface
Communication
Network
User command interface

In addition to the five groups referenced here, the NIST/ECMA Frameworks reference model contains services related to framework administration and configuration; these are included in the End-User section of the PSESWG reference model.

3.3. Place of the target system in the model

While the target system may be the same as the development system, there is no requirement that this be so. The PSE reference model therefore differentiates between the services available on the host machine used in the development of computer-based systems, and services on the target machine upon which the developed system will execute. Within the NGCR program, some of the details of target system functionality are described elsewhere. One source for these details is the 'Operating Systems Standards Working Group Reference Model,' June, 1990 [3]. Other services, in particular those relating to connection and monitoring of target system services to the development system, are part of this PSEWG reference model, and are included in the End-user services listed in Section 3.1.

4. Commentary

Developing a reference model for such a diverse and complex application area has been a major challenge from a technical, a political, and an organizational view point. In this section we discuss some of the challenges and describe the lessons we have learned while undertaking this work.

4.1. What is a reference model?

The PSESWG reference model describes from a purely conceptual view point what functionality

a PSE can be expected to provide. The very concept of a 'reference model' is still far from being a well accepted notion and can therefore be a great source of confusion. In general, people are much more familiar with physical models or, at best, with (conceptual) models of a physical implementation such as architectural models. However, the aim of this PSEWG reference model is explicitly not to define a physical architecture.

Evidence of this confusion can be seen in a number of presentations, computing articles, and government acquisitions where a purely conceptual model for PSE frameworks is being used either to define a PSE architecture or as a basis for 'conformance'.

One of the important lessons from our work on the PSESWG reference model has been the importance of providing a clear and concise discussion of the role of our reference model. The separation of the 'actual' world from the "conceptual" world has been very carefully made (see *Fig. 5* below), and helps to provide a clearer understanding of the role of this work.

Our hope is that the concise definitions provided by the PSESWG reference model will help people in making evaluations of actual PSE products. In addition, the discussion generated by the POSIX, NIST/ECMA, and PSESWG reference models will provide the basis for a deeper understanding of the role and importance of 'reference models' and highlight the need for additional models (e.g. architectural models) in the PSE domain.

4.2. Conceptual models vs. actual environments

Since the reference model is conceptual as opposed to actual, the service descriptions tend neatly to partition the functionalities of a PSE. When an actual environment is examined, however, these neat conceptual groupings are seldom found. Real software components span various service groups, with many components considered to be end-user tools also providing capabilities properly regarded by the reference model as framework services. The likelihood of this functional overlap is the reason that a conceptual model is necessary: one of its principal values is that it provides a common conceptual basis against which to examine many different environment implementations. *Figure 5* illustrates the

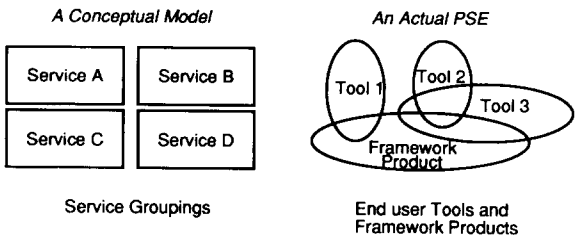


Fig. 5. Relationship between conceptual and actual worlds.

distinction between conceptual service descriptions, having no duplication of functionality, and a set of actual software components, many of which may overlap in their functional capabilities. As the figure shows, tools may duplicate other tools' functionality, and a tool often provides both framework and end-user services.

Note that even if actual environments show this mixing of framework and end-user functionality, it is nonetheless true that framework services tend to be a relatively fixed set of infrastructure services found in most environments, regardless of domain or tool content. Some of these ideas have been further explored in [1].

4.3. Rationale for the groupings in the model

In the widest sense, all users of the computer system are ultimately participating in project execution. However, the reference model distinguishes end-user services as those that are directly related to project execution. Using the example previously cited, i.e. tool installation vs. engineering activities, installing a tool clearly can facilitate the eventual engineering process. However, services specifically related to tool installation are conceptually different enough from services that directly support high-level engineering activities that the reference model considers the classification of tool installation appropriately as a framework service and not as an end-user service. There are other criteria by which services are grouped in the reference model. Often a collection of services provides the functionality needed to support a common resource. For example, there are twenty-one services in this reference model related to accessing data objects in a common repository. These services are all considered part of the Object Management Services group. Since these services are related by creating, accessing, manipulating and using objects

from a repository, their classification as a single group allows for a better conceptual understanding of the requirements imposed on any realization of these services and ultimately on any standards that address these services. Another motivation for grouping some services together might be the roles individuals take in using them. Thus, the activities that go into designing and producing executable source programs will use services that are grouped under the heading of Software Engineering. In this case, the group is determined by the users of the service rather than the management of a common resource. The boundary between service groups, particularly the boundary between end-user and framework services, is a dynamic one that changes over time. There is a general tendency for greater functionality to be gradually assumed by the underlying framework. For instance, historically most operating systems have included a directory structure and file system for data storage; a relational database is only occasionally added to a basic operating system. In the future, however, relational database functionality may be part of every operating system. It is precisely this growth of functionality that leads toward the notion of 'framework,' in contrast to the notion of 'operating system.'

4.4. Distinguishing services from tasks

A fundamental distinction that is being made in the reference model is between a service provided by a PSE and the service consumer in the form of a task. For example, a compile service may be used as part of the compilation task. Such a distinction is useful in that the reference model itself describes services without being too closely tied to particular tasks or processes. Hence, we can describe many different tasks in terms of their required PSE support services.

Unfortunately, despite our efforts to make the distinction between service and task a clean one, there are a number of areas where such a distinction becomes difficult to make. Examples of these areas include acquisition management, logistics, and quality assurance. In each of these cases it is difficult to distinguish the services expected from a PSE in support of the task from the details of the task itself.

In this reference model we have attempted to make a distinction between service and task wherever possible, but also recognized he need to accept a blurring of this distinction in some areas.

4.5. Defining the scope of the reference model

Although there is general agreement in the community about what constitutes a PSE, it is our experience that this apparent agreement breaks down around the 'edges'. One direction of early conflict (which was fairly readily settled) hinged on whether or not to include the operating system and other similar 'low-level' services. Another more difficult one hinges on what constitutes an 'engineering domain' and how extensive the end-user services should be. For example, there is a significant difference between the following possible statements of the PSESWG RM scope:

- support for engineering of systems
- support for engineering of computer-based systems
- computer-based support for engineering of systems
- computer-based support for engineering of computer-based systems.

The last one, which is the one currently being used in the PSESWG RM, limits the services of interest to automated ones for engineering (which includes life-cycle support as well as development) of hardware and/or software systems. The first one, on the other hand, could lead to the inclusion of activities as well as services in all engineering domains (including, for example, chemical engineering and ship hull design).

The accompanying criterion we have used is to ask ourselves whether a normal project engineer (which includes managers) would expect to see the service available from his/her project's fully-populated PSE. This has helped considerably in cutting down the number of engineering domains that it is reasonable to try to cover in a PSE reference model.

Another concept of PSE scope we have had to handle is the idea that, if it is automated, then it belongs in the PSE. This is very seductive, as we would all like to describe the 'perfect PSE' that would provide automatic support for all sorts of reasonable project activities. But it is important that this reference model describe something that is relevant to the current marketplace and so is identifiable to and comfortable for those who

participate in it. Confining the scope of the reference model as we have strengthens its relevance to the community to which it is addressed and thus increases the prospect that the concepts it embodies can be realized.

4.6. Moving from services to interfaces

The end goal of PSESWG is to select standards that facilitate the assembly of a PSE from COTS tools. This reference model is therefore a step on the way to achieving this goal. Hence, we must consider the utility of the reference model in helping us to select those standards.

Our intention in defining this reference model is that it helps us to identify and understand the interfaces within a PSE where standardization is both possible and desirable. Such interfaces are, of course, an architectural consideration, while this reference model is explicitly a purely conceptual device. Hence, the current focus of the PSESWG is to concentrate our efforts on an analysis of existing architectural models, using the language and service categories of the reference model as the common basis from which they can be described. By performing such an analysis the expectation is that common interfaces across a set of architectures will become apparent and that details of the selection of interface standards relating to those interfaces can be compared. A number of major CASE tool vendors and environment builders have already begun to use the reference model to analyze their products.

Concurrent activities in examining the requirements for PSE interfaces and in collecting descriptions of candidate PSE standards also contribute to the goal of finding common interfaces. We will soon be able to make our own selection of standards relevant to those interfaces with knowledge of which interfaces are commonly supported in available systems, and which selections have previously been implemented. This gives us the greatest opportunity to be compatible with those systems, and hence to make use of their existing investment in knowledge, experience, and products.

4.7. Validation and mappings

A key activity within the PSESWG work is the validation of the reference model through its application to a variety of actual PSEs. This is not as straightforward a task as we would like. There appear to be a number of reasons for this.

Firstly, the complexity of such PSEs means that performing a mapping activity is a non-trivial undertaking. Estimates of the effort involved range from one day to more than two weeks. Second, there are few people with a detailed knowledge of both the reference model and the system to be mapped. Both the reference model and the PSE product to be mapped require significant time and effort to be able to perform a detailed, useful mapping. This requires additional work on behalf of the person performing the mapping. Often, even obtaining such detailed information on a PSE can be difficult if one were not closely involved with the development of that PSE. Third, in comparing mappings performed by different people it is evident that detailed mapping guidelines, examples, and perhaps training would be required if there is to be any hope for consistency across the mappings as reference models for broad application domains must of necessity leave a number of degrees of freedom. It is the role of mapping guidelines to help a a mapper in deciding how to address this lack of precision. Currently, few such materials are available and hence a goal of PSESWG is to help produce any necessary materials.

4.8. Coordinating multiple, complex application domains

In areas such as operating systems, networking and database systems, reference models now exist that help us to understand, relate, and explain the services and their relationships (e.g. the POSIX reference model, the ISO Open Systems Interconnection model, and the ANSI/SPARC model respectively). These models have been defined (and refined) over an extensive period of time following a large amount of experience with these particular application areas.

In the PSE domain, however, there is considerably less experience to draw upon. While it is true that tools to aid software development have existed since the earliest machines, the advances toward comprehensive, life cycle support has been very recent. This immaturity, coupled with the extensive collection of technological components considered part of a PSE (encompassing, for ex-

ample, the three areas mentioned above), lead to severe problems.

One particular problem of note is that the technical and political convenience of building on existing reference models in the constituent domains of a PSE brings with it its own advantages and disadvantages. For example, while providing a useful starting point for our work, it has also meant that we have had to address a number of technical inconsistencies and omissions within and between the models. Where these have been found we then have the technical and political problems of organizing how these should be addressed and by whom. As with all reuse activities, reusing conceptual models has the problems associated with amalgamating various component parts over which we have little direct control.

Finally, the technical challenge of creating a valid reference model, applicable to any engineering environment, is an extraordinary one. To achieve this goal, there is need for breadth of knowledge covering many domains, in many areas of application; there is equal need for an overview of these multiple domains. Above all, the reference model needs a coherent and logical basis if it is to be intellectually acceptable throughout the software community. The model is the result of exhaustive debate on the terminology, content, and structure of the coherent and logical basis for modeling the abstract behavior of a PSE.

4.9. Working group attendees

Membership on all NGCR's working groups is voluntary, and meetings are open to all interested parties. The membership is therefore broad, but often has uncertain continuity from meeting to meeting. The working group is also structured as a consensus organization, and decisions tend to be made by informal acclamation rather than by formal vote [8]. There are also some competing constraints inherent in a joint project between government and industry. Since NGCR is essentially a government program, there are some mandatory considerations from a government point of view that participants from industry find unnecessary. These issues were not (and are not) insuperable, but represent an ongoing need for compromise. It is likely that such compromise will be a feature of any reference model that attempts to gain consensus across a wide spectrum of government and industry.

[8] Given an open membership, it is difficult to achieve any notion of eligibility of a voter.

4.10. Lack of standards agreement in the PSE community

Another area of difficulty lies in the complex and unstable area of software standardization. Many groups are working in many domains, and some of the developing standards are either overlapping or in conflict. For example, there is a great deal of debate in such areas as object management, data formats, and object-oriented methods. While these standardization efforts are conceptually distinct from the PSESWG effort, they nonetheless have an impact on the development of the reference model, since the intellectual basis for some of the model's conceptual services can often reflect a prejudice about a given standard. Given that this lack of agreement over standards in the PSE domain will continue for the foreseeable future, dealing with the overlap and conflict of standards will continue to be a significant problem.

5. Conclusions and future work

In this paper we have reviewed the main elements of a PSE reference model that we have defined as a necessary step toward the goal of selecting standards that will facilitate the assembly of a PSE from COTS tools. Producing such a reference model has been a major undertaking involving a great deal of resources. We believe, however, that this effort has been very beneficial to our PSESWG goals in a number of ways:

- it is a focal point for producing a common set of concepts, terminology, and issues that are an essential basis for making progress in a large, multi-organizational effort such as the PSESWG.
- it is a framework for categorizing existing and proposed standards and products as a necessary precursor to standards selection.
- it is a public document that illustrates our intentions to the PSE research community, attracting people to attend, comment, and contribute to our efforts.

Additionally, we also believe that our work has much wider implications for others working in the PSE area, and in the software engineering area in general:

- it is an example of the kind of reference material that must be developed in the area of PSEs to provide a deeper understanding of a number of the issues that need to be addressed.
- it is a usable, practical document that can help in the analysis and evaluation of complementary and competing PSE standards and products – a task in which many organizations require help and support.
- it is a demonstration of the effectiveness of leveraging the talents and experience of government, academia, and industry to produce useful results that are of benefit to each of these communities.

Looking to the future, the release date for Version 2.0 of the reference model is October 1993 [9] after which the document will be revised periodically. To aid the work by members of PSESWG, the reference model has been reviewed by members of other working groups, notably the NIST ISEE workshops, the Technical Group on Reference Models (TGRM) of the ECMA Technical Committee 33 (TC33), and several of the contributing experts of the international Portable Common Interface Set (PCIS) program. These persons have made many valuable contributions toward the final document. In addition to developing the reference model, PSESWG also supports other related activities. For example, the reference model is being used by PSESWG members in several mapping activities, making use of the reference model as a basis for examining actual environments. More are planned during the coming year, and a future document will detail the results of these mapping activities. Additionally, a catalog of available technology has been compiled and will periodically be updated.

PSESWG is now moving toward a second stage, which is to examine actual standards and products selected from the catalog of available technology. Two new teams of working group members have been formed, one of which is investigating standards and products related to framework services, and the other examining standards and products related to data interfaces. These two groups will examine as many of these items as is feasible. The result of these examinations will be formal characterizations of the important interfaces, as well as a list of candidate standards for these interfaces. PSESWG's final activity will be to make actual selections of interface standards, which will then be collectively listed in a single NGCR PSE standard. Accompanying such a list will be a document describing detailed considerations of the relationships, overlaps, omissions, and options that must be considered in using the collection of standards.

Finally, there will likely be other merits of the reference model in addition to its planned use by NGCR. Its use as a basis for a common set of concepts and terminology with which to discuss the PSE domain will be a very useful contribution to the whole PSE community. Similarly, the reference model has potential value in the study and analysis of tool integration and may help in characterizing tool capabilities. These are both areas for future research for which the reference model may prove to be of value.

[9] The reference model documents are available from any of the authors and in electronic form from the PSESWG archive. Electronic mail inquiries should be sent to 'psearch@nadc.navy.mil' with a subject line of "help".

Acknowledgements

A large number of people have contributed to the work of the NGCR PSESWG in general, and the production of the PSE reference model in particular. We gratefully acknowledge their contribution to the work reported in this paper.

We also thank the internal SEI referees of this paper for their very valuable comments.

The SEI is sponsored by the US Department of Defense.

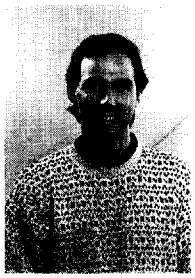
References

[1] A.W. Brown and P.H. Feiler, *An Analytical Technique for Examining Integration in a Project Support Environment*, Fifth Symp. on Software Development Environments, ACM (Dec. 1992).

[2] *Reference Model for Frameworks of Software Engineering Environment*, National Institute for Standards and Technology, Special Publication Number 500-201 (Dec. 1991).

[3] *Reference Model for Embedded Operating Systems*, NGCR Operating System Standards Working Group (June 1990).

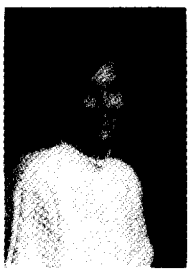
[4] *Draft Guide to the POSIX Open Systems Environment*, P10003.0 (June 1992).
[5] *NGCR Reference Model for Project Support Environments*, NGCR Project Support Environments Working Group, Version 1.0 (Feb. 1993).



Alan W. Brown is a member of the technical staff within the CASE Environments Project at the Software Engineering Institute, Carnegie Mellon University. Prior to the SEI, he was a lecturer in the Computer Science Department at the University of York, England and previously a research associate at the University of Newcastle upon Tyne where he received his PhD. for his work on the Aspect Integrated Project Support Environment (IPSE) Project. He has published numerous papers and three books: *Database Support for Software Engineering, Object-Oriented Databases and their Application to Software Engineering*, and *Software Engineering Environments*. His current research interests include software engineering environments and CASE tool integration.

David Carney worked on the development of the Ada Integrated Environment (AIE) at Intermetrics before joining the Institute for Defense Analyses (IDA) in Washington DC. There he participated in the Common APSE Interface Set (CAIS) and its revision, CAIS-A, and assisted in coordinating the SEE work that forms part of the Software Technology for Adaptable, Reliable Systems (STARS) program. Dr Carney is also a key player in both the NIST working group that has contributed to the NIST/ECMA reference model, and is chair of the reference model sub-group of the U.S. Navy's PSESWG. Dr. Carney is continuing his work on SEE standards and products as a member of the CASE Environments Project at the Software Engineering Institute.

Peter H. Feiler is Director of the Software Engineering Techniques Program at the Software Engineering Institute where his interests include software engineering environments, configuration management, software process support, development of large software systems, and software architectures. Prior to joining the SEI in 1985, he was a research scientist and group leader with Siemens Corporate Research and Technology Laboratories in Princeton NJ, which he joined in 1980. Feiler earned his Vordiplom in mathematics and computer science from the Technical University in Munich, West Germany. He received his doctorate in computer science from Carnegie Mellon University for his work on the Gandalf project. Feiler is a member of ACM, and the IEEE Computer Society.



Patricia Oberndorf received a BS in Computer Science from Oregon State University in 1971 and an MS in Computer Science from the University of California at San Diego in 1974. She has been active in environments research for over 15 years. She led the effort to define the Common Ada Programming Support Environment (APSE) Interface Set (CAIS) in the 1980's and has participated in international efforts as well. Tricia is currently the civilian co-chair of the Project Support Environment Standards Working Group (PSESWG) of the Navy's Next Generation Computer Resources (NGCR) program, which began meeting in February 1991.

Marvin V. Zelkowitz is a professor in the Computer Science Department and Institute for Advanced Computer Studies at the University of Maryland, College Park. He has a long-standing faculty appointment with the Computer Systems Laboratory of the National Institute of Standards and Technology where he works on software engineering environment standardization issues. His research interests include environment design and implementation and program complexity modeling, measurement and experimentation.