NISTIR 5487

Making Sense of Software Engineering Environment Framework Standards

Barbara Cuthill

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Bldg. 225, Rm. B266
Gaithersburg, MD 20899

NIST

NISTIR 5487

Making Sense of Software Engineering Environment Framework Standards

Barbara Cuthill

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology
Bldg. 225, Rm. B266
Gaithersburg, MD 20899

May 1994

Abstract

The purpose of software environment framework standards and specifications is to enhance tool portability, interoperability, and integration by creating public interfaces to functionality incorporated into the framework. If tools can access framework services in predictable ways, tool vendors can take advantage of these services to avoid duplicating services and concentrate on the unique functionality of the tool. This describes the functionality and integration support supplied by selected set of software environment framework standards and specifications with respect to common models.

1 Introduction

There is a growing interest in moving common functionality out of tools into the software environment framework allowing tool vendors to concentrate on the specialized features of their products and buyers to integrate tools easily into environments. Multiple standards and specifications for interfaces to common facilities for data, control and presentation integration in environments, especially software engineering environments (SEE), have begun to address these needs. This leads to the problem of determining which specifications are compatible and can coexist within the same environment. This article describes the functionality and integration support provided by eight environment framework standards and specifications with respect to common models. The selected framework standards and specifications follow:

- The International Standards Organization's Portable Operating System Interface (POSIX), Part 1: System Application Program Interface [25]
- European Computer Manufacturers Association (ECMA) Portable Common Tool Environment (PCTE) Standard[1] [21-23]
- Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) [18]
- Object Database Management Group's (ODMG) Object Database Management Standard (ODBMS)[20]
- Electronic Industry Associates (EIA) CASE Data Interchange Format (CDIF)[2] [1-15]
- American National Standards Institute (ANSI) Draft Messaging Standard[3] [26]
- MIT X Consortium's X Window System [31]
- Open Software Foundation's (OSF) Motif [29]

Even though these standards are associated with POSIX platforms, they use different notations, are in different stages of the standardization process, and meet different perceived needs. This article describes the current versions of these standards and specifications with respect to common models to examine the implications of these differences, the relationship of the standards and specifications to each other, how each supports integration, and why a SEE might combine these standards and specifications.

[1]ECMA submitted PCTE for ISO fast track standardization in Sept. 1993, and PCTE should become an ISO standard by fall, 1994.

[2]The CDIF committee will send completed standards to ISO/IEC/JTC1/SC7/WG11 for progression in ISO.

[3]CaseCommunique and Case Interoperability Alliance originally developed this standard and submitted it to ANSI X3H6.

The goal of developing SEEs is to create more than computer aided software engineering (CASE) tool collections. A SEE should *integrate* user selected CASE tools by supplying a framework supporting the smooth transition and exchange of control and data among tools in the same or different phases of the software development life cycle to support a defined software development process. One mechanism supporting tool integration is the use of standard interfaces to framework services.

No one standard provides an interface to all the needed framework capabilities, and the selected standards are not an authoritative list, only a starting point. Industry needs to choose and adapt sets of standards to form profiles defining how specifications provide an interface to an environment framework. The information gained from examining the selected standards and specifications relative to common reference models and the implications of the identified differences, overlaps and incompatibilities among the standards and specifications can support this profiling effort.

2 The *Reference Model for Frameworks of Software Engineering Environments [27]*

This article describes the functionality in each of the selected SEE framework specifications by comparing each specification to the Reference Model for Frameworks of Software Engineering Environments (Framework RM)[27]. The National Institute of Standards and Technology (NIST) Integrated Software Engineering Environment (ISEE) Workshop and the European Computer Manufacturers Association (ECMA) developed the Framework RM to define the capabilities of SEE frameworks. The Framework RM defines a list of capabilities (termed *services*) that a complete SEE framework might contain. The Framework RM divides these definitions into seven groups of related services. Framework services can support tool integration by supplying consistent services to the tools. Appendix A lists the Framework RM service definitions. Mapping a specification to the Framework RM can position the specification relative to a base set of SEE framework capabilities and help to define its relationship to other SEE framework specifications.

3 Integration

One goal of using a SEE framework is to support tool integration. This article describes each specification in terms of the possible support it might supply for tool integration in terms of integration categories and levels. *Integration categories* describe areas in which CASE tools can cooperate. *Integration levels* describe the extent to which CASE tools cooperate in a particular integration category. Since there is no accepted set of definitions or reference model for integration, this section provides definitions to clarify this article's discussion.

3.1 Integration Categories

There is general agreement that integrated tools should exhibit the same "look and feel," share information, or invoke each other when appropriate [32, 33, 36, 38, 44, 45, 46]. These capabilities correspond to the generally orthogonal categories of presentation, data and control integration [35, 36, 44, 46]. For example, two tools can simultaneously exhibit extensive presentation integration if their interfaces follow the same style guidelines and weak data integration if they exchange only unformatted text files. While other integration categories exist (e.g., [35, 44]), this article focuses on presentation, data, and control integration because these have received widespread acceptance as fundamental to leveraging the combined capabilities of tools in an environment [35, 36, 44, 46]. The definitions for the categories used in this article follow:

Presentation integration imposes consistency on the tools' user interfaces. Consistent user interface conventions reduce the burden on the user to learn and maintain several sets of interface commands and allow the user to concentrate on the unique functionality of each tool.

Data integration allows users to effectively employ multiple tools on a data item or a group of connected items. An environment could support data integration through pipes, common interfaces, or a shared repository. Sharing information implies that the tools use a common format for locating and interpreting information.

Control integration allows the use of multiple tools on a single project. Mechanisms for control integration usually support communication of notifications and requests among tools about events such as the starting or ending of tasks. Control integration mechanisms can also support combining tool functionality through remote procedure calls or other methods.

3.2 Integration Levels

This article uses Brown and McDermid's [36] integration levels combined with Wasserman's [46] progression of integration capabilities to describe the extent of the integration support provided by a SEE framework specification. The first four of Brown and McDermid's [36] proposed integration levels are carrier, lexical, syntactic, and semantic. Tools which can independently use the same resource exhibit *carrier* level integration (e.g., many simple POSIX tools can operate on the same ASCII file). Tools which use common formatting conventions exhibit *lexical* level integration (e.g., specific messages to trigger tool operation). Tools which use a common set of rules governing cooperative use of a shared resource exhibit *syntactic* level integration (e.g., a cut and paste mechanism allowing the tools to cut and paste information among their windows). Tools which use a common set of definitions for the meaning of common structures exhibit *semantic* level integration (e.g., the user commands for the cut and paste commands in the previous example). Table 1 provides example capabilities for the different integration categories and levels. These definitions are the result of mapping Wasserman's [46] sequences of specific integration capabilities to Brown and McDermid's abstract integration levels [36] and extending Wasserman's definitions as needed.

Table 1: Integration Levels and Capabilities

Integration Category	Carrier	Lexical	Syntactic	Semantic
Data	file transfer	using shared files	use of the same database/object base	use of the same metadata
Control	remote procedure call	triggers	message servers	agreed messages
Presentation	use of the same window system	use of the same window manger	use of the same toolkit	standard semantics for the toolkit

4 Reference Model Mappings

The first stage in developing a profile is to understand the functionality of each proposed standard or specification. This section examines the functionality and integration support of the selected specifications by mapping them to the Framework RM services. Identification of the services (see Appendix A for service definitions) that a specification supplies begins the process of identifying overlapping and complementary services supplied in a set of specifications.

The selected standards each supply a different set of framework services combining in different specifications. OSF intended Motif to supplement X Windows making the combination of services Motif requires and X-Windows supplies compatible. There is an X Windows toolkit available to implement the features needed to support the Motif style guidelines. The X3H6 draft messaging standard and ODBMS developers were aware of CORBA and worked to make these standards potentially compatible with CORBA. The X3H6 draft messaging standard requires services that CORBA generally supplies; however, ODMG did not design ODBMS to meet CORBA's repository requirement but as a database interface compatible with CORBA. Neither of the repository specifications discussed supplies exactly the services CORBA requires[4]. Similarly, it is unclear if PCTE schemas or ODBMS can implement CDIF meta-models.[5] All these standards and specifications were intended for products running on POSIX compliant platforms.

4.1 POSIX

Functionality Provided -

The Portable Operating System Interface, Part 1 (POSIX) provides a standard interface definition to key operating system services. These operating system services manage the low level resources that applications may need to access or manipulate. These resources include file systems, attached hardware devices, and operating system processes. POSIX provides an

[4]The OMG PCTE SIG is developing extensions to PCTE which should improve its compatibility with CORBA.

[5]ISO/IEC JTC1/SC7/WG11 (Software Engineering Data Description and Representation), CDIF and ECMA TC33 TGDI are proposing to define PCTE schemas for CDIF meta-models.

interface to services for managing and communicating with these and the underlying environment. Table 2 provides a mapping of POSIX to the Framework RM's operating system services. This mapping draws on the PSESWG mapping [28] POSIX provides an interface to the services that other environment standards and products can use.

Table 2: Summary Mapping of POSIX

Operating System Services	POSIX
OS Process Management	Extensive support for the creation, management, and termination of processes
OS Environment	Extensive support for querying and setting environment variables and process characteristics
OS Synchronization	not supported
Generalized Input and Output	Support for the creation, deletion and use of pipes and files as means of transferring information among processes
File Storage	Extensive support for creating, deleting and managing files and directories
Asychronous Event	Supports the generation and delivery of signals
Interval Timing	Supports explicit interval timing
Memory Management	Supports dynamic memory management
Physical Device	Management of modems, terminals, and printers as IO devices
OS Resource Management	Management of System Databases

Integration Supported -

POSIX provides a platform for minimally integrating applications in an environment. POSIX provides the possibility of access to common resources without restrictions on how applications will use those common resources. This access to common resources provides *carrier level integration* for the applications using them. Because POSIX defines a file system interface, tools can operate on the same files allowing for carrier level data integration. Because POSIX defines a signalling mechanism and pipes for processes to communicate, tools can exchange explicit signals supporting carrier level control integration. Because POSIX supports communication with terminals and the definition of user sessions and child processes, multiple tools can communicate with the user via the same terminal providing carrier level presentation integration. While POSIX, by itself, supplies very weak integration capabilities, it supplies a standard interface to the underlying resources needed for more extensive integration using other specifications.

4.2 PCTE

Functionality Provided -

The Portable Common Tool Environment (PCTE) standard merges operating system and repository functionality by providing an interface to services allowing tools to communicate, share data, and manipulate operating system processes, data, user groups, message queues and other environment components. PCTE manages processes, message queues and all other repository objects using the same repository operations. PCTE organizes the repository using schemas defined on the entity-relationship-attribute data model.

PCTE attaches security and integrity labels to all items in the repository. Entities, attributes and links can have security labels or separate locks. PCTE provides 11 types of locks for concurrency control and more than 20 types of access rights. These mechanisms allow PCTE to support complex security and integrity policies but also make PCTE complex to use and require substantial overhead on each element in the repository.

PCTE extends the repository interface to include control of operating system processes and limited support for tool communication. PCTE can start, monitor, and close operating system processes while treating the process as an object within the PCTE repository. For tools running as PCTE processes, PCTE supports triggers used to send notifications to specific executing tools and message queues and pipes for direct communication between executing tools.

In terms of the Framework RM, PCTE provides an interface to most of the object management and policy enforcement services and to some of the communication services. Table 3 summarizes the mapping to the object management service, Table 4 summarizes the policy enforcement services, and Table 5 summarizes the communication services. This mapping draws on several previous mappings of older versions of PCTE to various editions of the Framework RM [35, 37, 39, 41, 44] primarily Carney's mapping[37].

Table 3: Summary PCTE Object Services Mapping

Object Management Services	PCTE
Metadata	allows creation of user-defined schemas containing different types; enforces typing and schema constraints on repository; supports rich semantics for metadata
Data Storage and Persistence	stores persistent data in repository
Relationship	requires that all objects have a link to another object; controls relationships through metadata
Name	provides multiple alternate "names" or "pathnames" to reach an object; provides unique but temporary handles to objects
Distribution and Location	can manipulate workstations, volumes, and devices as any other objects in the PCTE repository; designed for distributed workstation environment
Data Transaction	are a class of activities; support for nested transactions
Concurrency	supports concurrency; provides 11 locking modes
OS Process Support	can start, monitor and close OS processes; treats OS processes as objects in the object base
Archive	supports archived objects; can assign objects to the contents of an archive; maintains metadata and links among archived objects
Backup	not covered
Derivation	supports the inheritance of attributes
Replication and Synchronization	can define or delete replicated objects; updates all copies of an object
Access Control and Security	provides rich syntax and semantics for defining security levels, projects, user groups and access rights; access control on all objects, links and attributes
Function Attachment	not covered
Common Schema	working schema provides the schema which is managing the data at any given time; working schema assembled from user- and pre-defined schemas
Version	supports the definition and manipulation of versions of objects
Composite Object	can define and manipulate composite objects
Query	can query object base
State Monitoring and Triggering	supports monitors and triggers and the creation, sending and deletion of notifications
Data Subsetting	provides views through schema definitions; treats schemas as filters; applies access restrictions to views
Data Interchange	can get files from and send files to other systems

As the mapping shows, PCTE provides a repository interface that allows for the creation, management and maintenance of metadata in the form of schemas which describe and manage the objects in and define views on the repository. The schemas describe all of the objects in the repository; although, the schemas can describe parts of the same objects from different points of view allowing subsetting of the repository. If the current schema (working schema) does not describe an object's type that object is not visible.

Table 4: Summary PCTE Policy Enforcement Services Mapping

Policy Enforcement Services	PCTE
Security Information	defines user groups, roles, access privileges and integrity levels
Identification and Authorization	provides authentication services
Mandatory Access Control	assigns confidentiality labels to objects and confidentiality clearances to users; enforces restrictions; controls information flow
Discretionary Access Control	sets access control lists; controls tool access to objects; has 20 access modes
Mandatory Integrity	assigns integrity labels to objects and clearances to users; enforces restrictions
Discretionary Integrity	see discretionary access control
Secure Exportation and Importation	requires security labels and access control lists on all objects; exports and imports labels; assigns labels volumes, workstations, devices etc.
Audit	maintains audit records according to audit criteria

The mapping to policy enforcement services demonstrates the substantial support available for these services in PCTE. These services are useful, but exact a price in the overhead on each object, link and attribute since PCTE maintains integrity, security and audit information about each element. Future extensions of PCTE should address this granularity issue.

Table 5: Summary PCTE Communication Services

Communication Service	PCTE
Data Sharing	via PCTE's object base which is managed through the definitions in the metabase
Interprocess Communication	PCTE processes can start, send messages to, stop or interrupt other PCTE processes
Networking	not covered (assumed to exist)
Message	message queues are PCTE processes; message queues can be reserved; some provided message types which the user can add to or extend
Event Notification	can send event notifications

PCTE supplies limited communication services. PCTE supports direct communication among tools running as PCTE processes but does not supply a general messaging facility. PCTE processes are the result of executing PCTE objects, and PCTE controls executing processes as it does other objects. In a full PCTE environment, there may be no processes running outside PCTE [45].

Integration Support -

PCTE supports the definition of schemas for describing and managing the repository contents requiring the storage of those schemas in the repository. Since new tools can use subsets of the available schemas available within the repository, multiple tools can access the same data from different viewpoints through the development or use of compatible schemas defining different views on the same objects. Tools taking advantage of the PCTE repository are at least *syntactic level data integration* and possibly *semantic level data integration* depending on their use of common schemas.

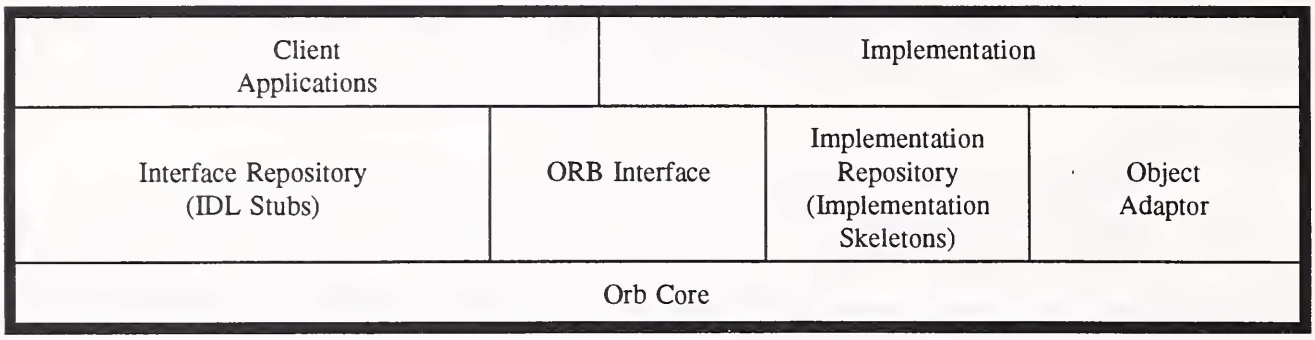
PCTE does have significant limitations on integration. PCTE cannot support tool integration through interface definitions. In PCTE, objects do not have interfaces (e.g., methods, functions) as recent object-oriented techniques use; however, there are extensions to PCTE in process which should address this. PCTE's triggers, message queues and pipes provide limited control integration which is available only to tools running as PCTE processes limiting the usefulness of these mechanisms.

4.3 CORBA

Functionality Provided -

The Object Management Group (OMG) defined the Object Request Broker (ORB) to provide interoperability among applications in heterogeneous distributed environments by supplying a mechanism for objects to make requests and receive replies[18]. In an object messaging model, objects send requests to other objects. The receiving object determines how to satisfy the request by invoking one or more of its interface services. The implementation of the service then performs the operation. In practice, a message routing mechanism such as the ORB selects the service implementation receiving the message on behalf of the designated object. For example, a message to a file object requesting it to display itself is routed directly to the editor which can display that type of file.

OMG's Common ORB Architecture (CORBA) specifies the infrastructure for supporting message passing among a variety of tools, and repositories in a distributed, dynamic environment. CORBA consists of several components, shown in Figure 1. CORBA uses the *Interface Definition Language* (IDL) to specify object types by specifying the operations in the object's interface. Client applications provide IDL descriptions of their interfaces. *Client stubs* are programming language bindings of these IDL descriptions. Tools can compile links to and invoke the client stubs. Alternatively, tools can obtain operation descriptions at runtime through an *interface repository* which provides persistent object interface descriptions that client applications can dynamically invoke. The ORB uses an *implementation repository* to identify the implementation supplying the requested service. The ORB uses *implementation skeletons* to call implementations of object methods. The *object adaptor* provides the ORB services to the implementations.



Client Applications
Implementation
Interface Repository (IDL Stubs)
ORB Interface
Implementation Repository (Implementation Skeletons)
Object Adaptor
Orb Core

Figure 1: CORBA Diagram

CORBA provides mechanisms for objects to invoke each other's operations. CORBA treats objects as opaque by requiring that tools exchange object references or handles. Interfaces either client stubs or from the interface repository provide access to an object's contents. Table 6 provides a mapping of CORBA to the Framework RM communication services drawing on an earlier mapping[43] done for NGCR PSESWG.

Table 6: Summary: CORBA Communication Services

Communication Service	CORBA
Data Sharing	can convert an object reference into a value that a client can store and use later to access the object
Interprocess Communication	via messages and events
Networking	assumed to exist
Message	can create, send, and delete messages; message passing can be synchronous, deferred synchronous or asynchronous
Event Notification	can raise exceptions or send requests that do not require a reply

CORBA clearly provides the messaging and event notification services supporting interprocess communication; however, CORBA assumes the existence of network capabilities and provides limited data sharing.

The ORB needs minimal repositories (i.e., the implementation and interface repositories) for its operation. The environment may provide these as separate repositories, one repository, dedicated CORBA repositories or general purpose repositories. CORBA includes assumptions about supplied object management services and their interfaces but not about the supplier of those services. Table 7 lists the object management services that CORBA requires.

Table 7: Summary: CORBA's needs for Object Management Services

Object Management Services	CORBA
Metadata	define types of objects by their interfaces; type codes provide definitions of the structure of parameters
Data Storage and Persistence	interface repository stores object type definitions and interfaces
Relationship	containers create relationships between objects and modules containing objects
Name	Repository ID uniquely identifies objects to ORB; separate ID is dependent on "container" object or module; can use module ids to navigate namespace
Distribution and Location	objects can be distributed across a network
Derivation	inheritance and limited multiple inheritance allowed; inherited objects and interfaces can extend definition not restrict or overwrite it
Replication and Synchronization	can duplicate and release copies of objects and object references
Access Control and Security	provides the principal requesting the operation to the implementation carrying out the request; ORB does no checking itself
Function Attachment	interfaces can be associated with objects
Query	can query the Interface Repository
Data Interchange	provides connections for foreign object systems

Integration Supported -

CORBA defines a complete messaging environment infrastructure providing at least *syntactic level control integration* to the tools using it. CORBA supplies facilities for tools to exchange messages and to use operations but does not specify the syntax or semantics of the messages. By proving both a static interface through client stubs and a dynamic interface through the interface repository, CORBA allows tool developers to decide how to combine performance and flexibility for individual tools.

CORBA meets its goal of providing maximum control integration with minimum data sharing. Tools using CORBA publish externally information about operations that they supply while providing minimal information about the data structures they manipulate.

4.4 ODBMS

Functionality Provided -

The Object Database Management Group (ODMG), a consortia of leading object-oriented database (OODB) vendors have defined the Object Database Management Standard (ODMS) for incorporating OODBs into tools, environments and systems [20]. The goal of ODBMS is to extend the semantics of OO programming languages to transparently access OODBs. ODBMS database objects possess the usual characteristics of OO programming objects: associated methods, attributes, and inheritance.

To achieve its goals while maintaining compatabiltiy with other standards, ODMG extends existing standards. ODBMS includes an Object Definition Language (ODL) for OODBs compatible with CORBA's Interface Definition Language (IDL), and an Object Query Language (OQL) partially compatible with ISO Structured Query Language standard (SQL)[30]. ODBMS uses the current (relational) version of SQL. It is important for future harmonization work in the database community that ODBMS be compatible with the next (object-oriented) version of SQL.

The ODBMS standard supports a strongly typed OODB controlled through metadata defined in ODL. ODBMS supplies predefined types which form the root of the type hierarchy, all user-defined types ultimately inherit from these predefined types. The same operations apply to user and pre-defined types. ODBMS supplies the ODL, OQL and OML (Object Manipulation Language) operations in bindings to C++ and Smalltalk to support portability of and transparent access from applications in these languages. Table 8 provides a summary of the ODBMS mapping to the Framework RM's object management services.

Table 8: Summary ODBMS Mapping

Object Management Services	**ODBMS**
Metadata	can define object and interface types; enforces typing
Data Storage and Persistence	stores persistent and non-persistent data in repository; requires specification of object lifetime on creation
Relationship	defines relationships between object types as traversal paths; maintains referential integrity
Name	uses unique ID for each object and flat namespace
Distribution and Location	supports defining site for an operation
Data Transaction	supports nested transactions; restricts object access to transactions
Concurrency	supports pessimistic concurrency control
Derivation	supports inheritance and multiple inheritance
Function Attachment	defines behavior of types; enforces on instances
Common Schema	permits the definition of a common schema; provides predefined schema
Composite Object	permits structured and unstructured collections of one type
Query	provides language for querying the object base (OQL); includes select and select-from-where

Integration Supported -

Because the ODBMS specification builds on ISO SQL [30] and CORBA's IDL [18], its integration support builds on that available from SQL and IDL. ODBMS supports schema design moving it into the area of *syntactic level data integration* by establishing ODL as a schema definition language and by building on SQL for a common query language.

While ODBMS supports data integration through schema definitions, there is no mechanism for enforcing the schemas on the general OODB or repository. The tool developers must compile these definitions within the tools. This is a significant drawback because developers must agree on the schemas in advance making it difficult to add new tools using the same schemas to the environment.

4.5 CDIF

Functionality Provided -

The CASE Data Interchange Format (CDIF) is a set of standards for describing the data imported and exported among CASE tools and repositories. CDIF's component standards describe interchanged data from the following viewpoints:

- the physical representation of the data (*encoding*)
- the grammar and structure of an exchange format (*syntax*)
- the datatypes or schemas (*meta-models*) in areas of specialization (*subject areas*)

The *meta-meta-model* is the set of rules for defining meta-models in CDIF. The meta-meta-model includes the minimum information needed to describe the meta-model defined for each of the CDIF subject areas. The CDIF committee has deliberately kept the meta-meta-model, like the rules for defining encodings and syntaxes, as simple as possible. The meta-meta-model includes meta-objects and meta-relationships which a subject area can refine into a meta-model.

The CDIF meta-models in specific subject areas define a meta-model or metadata for that area. The CDIF committee has or is planning to define meta-models in a number of subject areas including the following:

- Foundation [6]
- Data Definition [3]
- Data Flow Model [4]
- Physical Relational Data Base [7]
- Presentation Global [10]
- Common [2]
- Data Modeling [5]
- State/Event Model [11]
- Presentation Location and Connectivity [8]
- Project Planning and Scheduling [11]

All the subject areas define views of the Integrated Meta-Model (IMM); therefore, subject areas can provide overlapping views of the same elements. While CDIF users can define new subject areas consistent with the existing ones, they cannot alter the existing subject areas for conformance and portability reasons. CASE tools can use one or more subject areas to define a data model, but not exclude subject area elements from the model. All the current subject areas build on the foundation and common subject areas.

CDIF both supplies and requires several Framework RM object management services for the management of *metadata.* The CDIF standards require that a system supply some functionality in order to use CDIF. Table 9 maps CDIF to the object management services.

Table 9: Summary: CDIF Object Management Services Mapping

Object Management Services	CDIF
Metadata	uses metadata management and creation services supplied in repositories or tools
Relationship	uses definitions of relationships
Derivation	uses a generalization/specialization hierarchy, limited multiple inheritance and inheritance of relationship types
Function Attachment	can attach "usage" or operations to an entity
Common Schema	defines meta-models in several subject areas
Composite Objects	supports definition of "collectable" meta-objects
Data Subsetting	uses different views of data
Data Interchange	defines a format exchanging data and meta-data

Integration Supported -

By using bindings of CDIF meta-models to specific syntaxes and encodings, tools can achieve *semantic level integration* because CDIF meta-models, syntaxes, and encodings supply the necessary definition of the structure and meaning of exchanged data. One possible binding is to PCTE schema definitions. ISO committee JTC1/SC7?WG11 and the CDIF Committee are investigating the use of PCTE schema definitions of the CDIF meta-models as working schemas for structuring the PCTE repository. While this is theoretically possible, it has not yet been done, and there is some concern that the CDIF committees designed a mechanism for data exchange without considering its possible uses as a repository schema definition[6]. There may be different tradeoffs associated with a repository scheme than with an exchange format. CDIF could provide a set of standard definitions for meta-models or schemas supporting semantic level data integration.

4.6 X3H6 Draft Messaging Standard

Functionality Provided -

The ANSI standards committee X3H6 (CASE tool integration methods) has recognized that for tools to effectively use messages requires agreement on the infrastructure to exchange messages, the semantics of the messages, and the syntax for recording the messages. The X3H6 Draft Messaging Standard (originally supplied to ANSI X3H6 by Case Communique and Case

[6]Minutes of ISO/IEC/JTC1/SC7/WG11 meeting in Orlando, FL in January, 1994.

Interoperability Alliance) defines a set of abstract messages (termed *servicegrams*) representing agreement on message semantics. The draft standard defines a set of requirements on an environment exchanging bindings of these servicegrams. The servicegrams are an abstract representation of the semantics of messages; X3H6 has not defined bindings for the messages in any specific syntax. Representation of the messages in a specific syntax has been left for the vendors and consortia until a common or stable representation has emerged. Agreement on the message semantics should make agreement on the syntax easier. In future, the authors of the standard hope that tool vendors will be able to reference on the tool's label, the messages the tool sends and the messages it accepts.

The draft standard presumes but does not require an object-oriented model of messaging, similar to CORBA's approach, and places specific requirements on the messaging infrastructure. These requirements are the minimal assumptions built into the messages. These assumptions on the messaging environment follow:

- It uses an asychronous or deferred synchronous routing mechanism.
- It supports request messages which require a reply (if the tool cannot supply a reply then the environment will).
- It supports notification messages which do not require a reply.
- An object receives multiple messages from the same source in the order they were sent.
- The environment will at least deliver messages to all running tools that have registered to receive that type of message.
- The message routing mechanism may use the name of the message, the type of the primary parameter and the primary parameter itself to route the message.

A servicegram specifies all the information that a developer needs for writing a message corresponding to that servicegram in a messaging notation and to define the requested service. X3H6 provided a template for the servicegrams which follows:

- **Type** - Request or notification
- **Name** - Unique identifying name, describes the expected operation performed on the primary parameter or state of the primary parameter
- **Primary Parameter** - Artifact that the servicegram concerns
- **Description** - English language description of the purpose of the servicegram
- **Required Parameters** - Auxiliary parameters to the primary parameter
- **Optional Parameters** - Allowed on request messages, may have a default value
- **Error Parameters** - Parameters describing returned error conditions
- **Preconditions** - Expected state of the environment when the servicegram is sent
- **Postconditions** - Expected state of the environment on the successful completion of the services invoked by the servicegram
- **Failure Conditions** - Conditions under which the servicegram will return an error

The postconditions and failure conditions define the expected responses of the tool to a specified range of inputs. These inputs are a mix of the environment or context variables and the parameters. For situations in which the inputs or context variables are outside of this range, the behavior of the tool in response to the servicegram is undefined. This leaves a range of inputs for the extension of the tool behavior beyond the minimal requirements of the servicegram.

Parameters are complex entities. There are four categories of parameter: primary, required, optional and error. The primary parameter is singled out for routing purposes since the message is sent to this object. The parameter template records the semantics of what each parameter is and how the tools will use it. The template describes the following attributes:

- **Name** - Name of the parameter
- **Description** - English language description of what the parameter represents
- **Type** - One of the specified types for parameters
- **Invariant** - Qualifications on the value of the parameter
- **Constraints** - Any constraints on the value of the parameter
- **Default Values** - Values to be used in place of optional parameters

There are servicegrams for the following functional areas:

- Software Analysis and Design
- Common (creation/deletion etc.)
- Debug
- Edit
- Static Analysis
- Window
- Build
- Configuration Management
- Documentation
- Process Management - Agenda
- Process Management - Enactment

The message sets contain functionally related messages. There is no conformance requirement or expectation that an application would be able to process all of the messages in any one group. Conformance is at the level of the individual servicegram so that a tool can define a set consisting of any combination of servicegrams representing its interface. The standards developers expect that an application will supply lists of the request messages that the application can respond to, notification message that it will process and messages that it will send. This labelling will allow environment integrators to recognize if the application supplies needed services for a particular environment.

Integration Supported -

This draft standard provides one of the necessary pieces for *semantic level control integration* by providing agreement on the contents of messages. While the standard defines requirements on the environment, in the same way that CORBA defines requirements on a repository, it does not define the environment. The authors of the standard used an object-oriented model of messaging to remain compatible with CORBA which is one mechanism for supplying the messaging environment infrastructure. The servicegrams are a first step to building commonly recognized messages for use on such an infrastructure. Developers still have to agree on bindings for the servicegrams before there are commonly recognized messages available.

4.7 X Windows

Functionality Provided -

The X Windows System provides a single widely accepted interface to workstation display systems. The goal of the X Windows System is "to provide a network-independent and vendorindependent operating [display] environment for workstation software." [41] An application running on X can use any vendor's display hardware through the same interface, and control multiple displays across a local area network. The wide acceptance of X Windows has meant that tool developers do not have to rewrite their tools for different display hardware; a version of X exists for almost every machine.

The X Windows System provides a layered set of components. Figure 2 illustrates the structure of those components. The Base Window System provides a common interface to the display hardware (vendor-independence). The X Network Protocol is the interface to the Base Window System for all applications, window managers, programming language interfaces, or toolkits running on X. The network protocol can operate within or between CPUs providing network transparency. The X Window Manager organizes and arranges windows according to a user defined policy. A window manager is crucial to allowing the display of multiple applications. The X Windows System provides a library of C interface routines which access the Base Window System through the network protocol. Xlib provides access to a large, flexible set of primitives for access and control of input and display features. There are alternative interface libraries for other programming languages. The toolkit provides these features in combination to gain more complex display components.

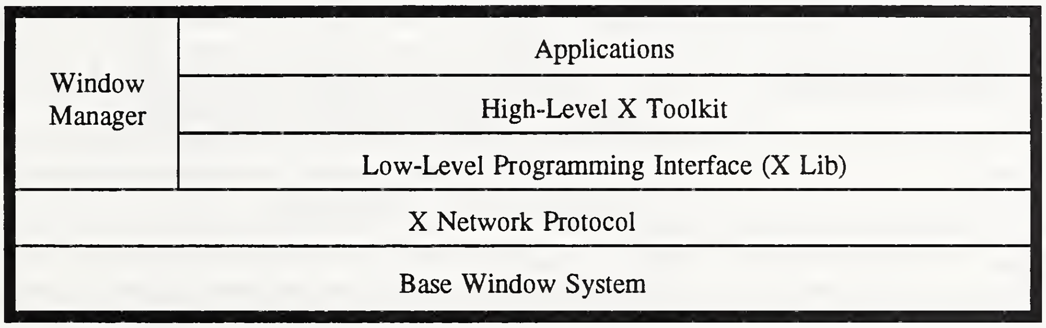


Figure 2: X Windows System Structure[41]

The X Windows System provides most of the capabilities of the Framework RM User Interface Services as Table 10 shows. This mapping extends the mapping from the NGCR PSESWG Progress Report [43].

Table 10: Summary mapping of X Windows to the User Interface Services

User Interface Services	X-Windows
User Interface Metadata	Xlib, Xt Intrinsics, X Widget Sets provide the connections to the display hiding increasing amounts of the low level details of the implementation and making the establishment of conventions possible
Session	X-Windows supports setting defaults and startup information for configuring the user interface
Text Input	supports user input through text
Dialog	sends "events" to client applications when the user provides input and supports the response of the client application
Display Management	provides a window manager to control the size and location of windows on the screen; Xlib provides C functions for interacting with the window manager
Presentation	Xlib and Xtoolkit define, create, delete, and manage display objects
User Interface Security	not covered
User Interface Name and Location	X Windows servers can execute client operations and control displays on other workstations in the network
Internationalization	X windows supports changing the standard character set and conventions used in the display
User Assistance	not covered

The X Window System defines its own messaging system on top of the network to provide network transparency. Table 11 maps the X Window System to the Framework RM Communication Services.

Table 11: Summary mapping of X Windows to the Communication Services

Communication Services	X-Windows
Interprocess Communication	Can exchange information among client processes
Message	can pass messages
Network	transparent to network
Event Notification	can pass event messages to client applications; typical mechanism driving applications is to wait until receipt of an event notice indicating the user has performed an action and respond to event

Integration Support -

X provides the mechanisms and network transparency needed for *syntactic level presentation integration.* Because X Windows has concentrated on defining as complete and as flexible a set of mechanisms as possible, X has been able to represent a wide range of policies and has achieved broad acceptance as an interface standard. The X consortium has specifically avoided the issue of defining policies for using those mechanisms.

4.8 MOTIF

Functionality Provided and Integration Supported -

Motif provides a set of user interface policies outlined in the draft standard [29] and implemented in the Motif toolkit[29]. The purpose of Motif is to provide a consistent style of user interfaces across multiple platforms in order to minimize user confusion and the time needed to understand new user interfaces. Consistency in the user interface allows tool developers and users to focus on the parts of the user interface accessing important or unique tool functionality.

Motif provides a toolkit for the X Window System. The X Window System provides the mechanisms to design standard interface toolkits combining features in expected ways. The Motif toolkit is a subroutine library of the features needed to implement a Motif style user interface. While a developer can implement the Motif style using other mechanisms, the Motif toolkit already has an interface to the necessary features.

The X Window System provided the means to specify the user interface metadata through toolkits; Motif actually specifies those user interface policies. Establishment of a common user interface policy corresponds to the User Interface Metadata Service of the Framework RM. Motif provides *semantic level presentation integration* on top of the X windows mechanisms by providing a specification for the semantics or look and feel of the user interface.

5 Combining Standards

5.1 Functionality

Section 4 contained mappings of the selected standards and specifications to the Framework RM. Table 12 summarizes these mappings. This summary lists the services supplied or required by each standard or specification identifying broad areas where relationships might exist. More work will help to understand the relationships among the specifications in the following circumstances:

(1) Multiple standards providing extensive (E) interfaces to the same service -

If there are two different versions of a service in an environment, system integrators may not know which to use. If both specifications must be part of the environment, this may be an important area for defining guidelines on using the services.

(2) A specification supplying a service that another specification requires -

The specifications may be a useful combination, and a more detailed mapping of the operations supplied and required can indicate if the two specifications are presently compatible or need some evolution or need guidelines on how to combine them.

(3) A specification requiring a service that no other selected specification supplies -

This illustrates a possible gap in coverage in the selected set of specifications.

(4) Lack of support for object-oriented technology (OOT) as indicated by failure to provide extensive coverage for both the derivation and function attachment services -

OOT typically supports inheritance among objects (derivation service) and attaching interfaces to objects (function attachment). If OOT is important then those services are important for the environment.

To understand the compatibility issues among a set of standards requires a detailed mapping of the operations defined in those standards, including their syntax and semantics, to the reference model. Developing more detailed mappings is important for identifying where standards and specifications may need to evolve for greater compatibility.

5.2 Integration

The support these standards and specifications provide for integration is mixed. While use of a common resource through a set of common services may avoid duplicate functionality, it will not automatically define common behavior. Common behavior requires agreed conventions on the use of the resource. For presentation integration, the X Windows System specifies mechanisms for the presentation system while Motif specifies a policy for using those mechanisms. While there are other specifications in the presentation area, X Windows and MOTIF have received widespread acceptance. For control integration, among the selected standards, CORBA specifies a messaging environment supplying the control integration infrastructure needed for a SEE, and the X3H6 draft messaging standard begins to define agreement on messages or conventions for that environment. There are other specifications in the control integration area, but this set has received considerable attention and endorsement[7]. For data integration, several communities have developed competing repository and database interface standards receiving different levels of market acceptance and use (e.g., SQL [30], IRDS [19], and CAIS [16]). Among the selected standards, both PCTE and ODBMS supply mechanisms for data integration while CDIF begins to define the conventions for describing data.

While this article has drawn distinctions among presentation, data and control integration mechanisms, the reality is not as well-defined. The Common Desktop Environment (CDE) [17] by defining an integrated desktop is adding messaging infrastructure to its environment moving from presentation into control integration. OMG has begun to address the data integration issues of locating objects referred to in messages and defining the interface and implementation

[7]Over 350 companies participate in OMG and over 100 companies participated in CASE communique and CASE Interoperability Alliance.

repositories. PCTE has always included triggers, pipes and message queues for control integration. Combining framework standards and specifications to support the smooth integration of tools requires recognizing how current standards and specifications support integration.

Table 12: Summary of Mappings[8]

Integration	Data			Control		Presentation		All
Reference Model Services	**ODBMS**	**PCTE**	**CDIF**	**CORBA**	**X3H6**	**X**	**Motif**	**POSIX**
Object Management Services	E	E	P					
Metadata	P	E	RE	RE				
Data Storage & Persistence	E	E		RP				
Relationship	E	E	RE					
Name	E	E						
Distribution and Location		E						
Data Transaction	E	E						
Concurrency		E						
OS Process Support		E						
Archive		P						
Backup								
Derivation	E	P						
Replication & Synchron.		P						
Access Control & Security		E						
Function Attachment	E			RE				
Common Schema		P	E					
Version		P						
Composite Object	E	P						
Query	E	E		RP				
State Monitoring & Trigg.		E						
Data Subsetting	P	E						
Data Interchange		P	RP					
Process Management Services								

[8]RE - Requires extensive support from this service; RP - Requires some functions from this service; E - provides extensive coverage of this service; P - provides partial coverage of this service; L - provides a very limited version of this service

Table 12: Summary of Mappings (continued)[9]

Integration	Data			Control		Presentation		All
Reference Model Services	**ODBMS**	**PCTE**	**CDIF**	**CORBA**	**X3H6**	**X**	**Motif**	**POSIX**
Communications Services								
Data Sharing		P	E	E				
Interprocess Communication		P				RE		
Network				RP		RE		
Message				E	RE	L		
Event Notification				E	RE	E		
Operating System Services								
OS Process Mgmt		P						E
OS Environment		RE						E
OS Synchronization		RP						P
Generalized I/O	RP	RE		RP		RE		E
File Storage		RE						E
Asychronous Event		RP		RE		RE		E
Interval Timing								P
Memory Management								
Physical Device		P				RE		E
OS Resource Mgmt.		RP						P
User Interface Services								
UI Metadata							E	
Session						E	RE	
Text Input						L	RE	
Dialog						E	RE	
Display Management						E	RE	
Presentation						E	RE	
UI Security								
UI Name and Location						P		
Internationalization						P		

[9]RE - Requires extensive support from this service; RP - Requires some functions from this service; E - provides extensive coverage of this service; P - provides partial coverage of this service; L - provides a very limited version of this service

Table 12: Summary of Mappings (continued)[10]

Integration	Data			Control		Presentation		All
Reference Model Services	**ODBMS**	**PCTE**	**CDIF**	**CORBA**	**X3H6**	**X**	**Motif**	**POSIX**
Policy Enforcement Services								
Security Information		E						
Ident. & Authentication		E						
Mandatory Access Control		E						
Discretionary Acc. Control		E						
Mandatory Integrity Control		E						
Discretionary Integ. Control		E						
Secure Obj. Exp. & Import.								
Audit		E						
Framework Admin. Services								
Registration		L		L				
Resource Management								
Metrication								
Sub-Environment								
Self-Configuration Mgmt.		L						
License Management								

6 Conclusions

The purpose of framework standards and specifications is to enhance tool portability, interoperability and integration by creating public interfaces to functionality incorporated into the framework. If tools can access framework services in predictable ways, tool vendors can take advantage of those services to avoid duplicating services and concentrate on the unique functionality of the tool. Use of common services can increase tool integration by supporting consistency in the behavior of tools using those services.

There is a significant difference in the maturity, acceptance and compatibility between the interfaces focusing on presentation integration and the interfaces focusing on control and data integration. Products for POSIX platforms today, generally, use the X Windows System and follow Motif or a similar style guideline. The general availability of X Windows on POSIX platforms allows developers and users to rely on its presence. Developers recognize the

[10]RE - Requires extensive support from this service; RP - Requires some functions from this service; E - provides extensive coverage of this service; P - provides partial coverage of this service; L - provides a very limited version of this service

usefulness of not having to recode the user interface for each new platform, and users appreciate not having to learn an entirely new interface for each tool.

There are no standards with comparable acceptance and availability supporting data and control integration. CORBA has received considerable support from vendors but is not yet a routinely supplied part of the platform; however, vendors are beginning to commit to supplying CORBA. The lack of consensus on the definition of needs for repository capabilities has prevented consensus on repository standards. Each repository standard or specification was developed in response to a different set of perceived needs. This lack of consensus has prevented tool vendors from relying on the presence of a repository.

Mappings of a set of standards to the Framework RM can provide a common terminology for describing the standards. With standards and specifications coming from a variety of communities to meet different requirements on the SEE framework, current standards and specifications, like the eight discussed here, do not share a common terminology or view of a SEE. Mappings are an important way of positioning the standards in the framework. These mappings can contribute to a better understanding of how the standards may interact within a SEE profile.

References

Standards and Other Reference Works

[1] CASE Data Interchange Format - Framework for Modeling and Extensibility. Electronics Industry Association, Interim Standard #107, 1994.

[2] CASE Data Interchange Format - Integrated Meta Model - Common Subject Area. Electronics Industry Association, Interim Standard #112 1994.

[3] CASE Data Interchange Format - Integrated Meta Model - Data Definition Subject Area. Electronics Industry Association, Interim Standard #113, 1994.

[4] CASE Data Interchange Format - Integrated Meta Model - Data Flow Model Subject Area. Electronics Industry Association, Interim Standard #115, 1994.

[5] CASE Data Interchange Format - Integrated Meta Model - Data Modeling Subject Area. Electronics Industry Association, Interim Standard #114, 1994.

[6] CASE Data Interchange Format - Integrated Meta Model - Foundation Subject Area. Electronics Industry Association, Interim Standard #111, 1994.

[7] CASE Data Interchange Format - Integrated Meta Model - Physical Relational Data Base Subject Area. Electronics Industry Association, Interim Standard #117, 1994.

[8] CASE Data Interchange Format - Integrated Meta Model - Presentation Location and Connectivity Subject Area. Electronics Industry Association, Interim Standard #118, 1994.

[9] CASE Data Interchange Format - Integrated Meta Model - Presentation Global Subject Area. Electronics Industry Association, Interim Standard #120, 1994.

[10] CASE Data Interchange Format - Integrated Meta Model - Project Planning and Scheduling Subject Area. Electronics Industry Association, Interim Standard #121, 1994.

[11] CASE Data Interchange Format - Integrated Meta Model - State/Event Model Subject Area. Electronics Industry Association, Interim Standard #116, 1994.

[12] CASE Data Interchange Format - Overview. Electronics Industry Association, Interim Standard #106, 1994.

[13] CASE Data Interchange Format - Transfer Format - Encoding.1. Electronics Industry Association, Interim Standard #110, 1994.

[14] CASE Data Interchange Format - Transfer Format - General Rules. Electronics Industry Association, Interim Standard #108, 1994.

[15] CASE Data Interchange Format - Transfer Format - Syntax.1. Electronics Industry Association, Interim Standard #109, 1994.

[16] Common Ada Interface Set. United States Department of Defense, MIL-STD-1838A, 1989.

[17] Common Desktop Environment: Functional Specification. X/Open, Ltd., 1993.

[18] Common Object Request Broker: Architecture and Specification, Revision 1.1. OMG Document Number 91.12.1, 1991.

[19] Information Resource Dictionary System. International Standards Organization, IS 10728, 1992.

[20] *Object Database Standard: ODMG-93*. Ed. by R. G. G. Cattell. San Mateo, California: Morgan Kauffman, 1993.

[21] Portable Common Tool Environment: Abstract Specification. European Manufacturer's Association, ECMA-149, Version 2, 1993.

[22] Portable Common Tool Environment: C Binding. European Manufacturer's Association, ECMA-158, Version 2, 1993.

[23] Portable Common Tool Environment: Ada Binding. European Manufacturer's Association, ECMA-162, Version 2, 1993.

[24] Portable Common Tool Environment: C++ Binding. European Manufacturer's Association, Draft, 1992.

[25] Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API). International Standards Organizaton/International Electronics Commission IS 9945-1, 1990.

[26] Proposed Draft Standard, Messaging Architecture. American National Standards Institute, Committee X3H6, 1994 (draft).

[27] Reference Model for Frameworks of Software Engineering Environments, Edition 3. National Institute of Standards and Technology, Special Publication 500-211, 1991.

[28] Reference Model for Project Support Environments. National Institute of Standards and Technology, Special Publication 500-213, 1993.

[29] Standard for Information Technology - X Window System Graphical User Interface - part 1: Modular Toolkit Environment (Draft). IEEE Working Group P1295.1, 1993.

[30] Structured Query Language. American National Standards Institute Standard X3.135-1992 and International Standards Organization IS 9075:1992.

[31] X Window System Protocol, Version 11. MIT X Consortium, 1988.

Additional Reference on SEEs, SEE Specifications, and Integration

[32] Arnold, John E. and Gerard Memmi. Control Integration and its Role in Software Integration. Nanterre, France: EC2, 1992.

[33] Brown, Alan W. An Approach Toward the Selection of Data Interface Standards. Draft Technical Report from Next Generation Computer Resources Project, 1992.

[34] Brown, Alan W., David J. Carney, Peter H. Feiler, Patricia A. Oberndorf and Marvin V. Zelkowitz. "Issues in the Definition of a Project Support Environment Reference Model" in *Annual Report of the SEI*. Pittsburg: Software Engineering Institute, 1993.

[35] Brown, Alan W., Anthony N. Earl and John A. McDermid. *Software Engineering Environments: Automated Support for Software Engineering*. New York, McGraw-Hill Book Co., 1992.

[36] Brown, Alan W. and John A. McDermid. "Learning from IPSE's Mistake" in *IEEE Software*, Vol. 9, No.2, March 1992.

[37] Carney, Dave. Mapping of PCTE to the PSESWG Reference Model. Software Engineering Institute, to be published, 1993.

[38] Chen, Minder and Ronald J. Norman. "A Framework for Integrated CASE" from *IEEE Software* Vol. 9, No. 2, March, 1992.

[39] Earl, Anthony. NIST Reference Model (1.0a) Mapping of ECMA PCTE(149). HP Laboratories, Bristol, England, 1991.

[40] Flecher, Tom and Jim Hunt. Software Engineering and CASE: Bridging the Culture Gap. New York: McGraw-Hill, 1993.

[41] Jones, Oliver. *Introduction to the X Window System*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1989.

[42] NIST Framework Model Mapping Guidelines, Version 1.2. (Draft) National Institute of Standards and Technology, 1991.

[43] Report on the Progress of the NGCR PSESWG. NADC, 1993.

[44] Thomas, Ian and Brian A. Nejmeh. "Definitions of Tool Integration for Environments" from *IEEE Software*, Vol. 9, No.2, March, 1992.

[45] Wakeman, Lois and Jonathan Jowett. *PCTE: the Standard for Open Repositories*. New York: Prentice Hall, 1993.

[46] Wasserman, Anthony I. "Tool Integration in Software Engineering Environments" from *Lecture Notes in Computer Science: Software Engineering Environments* ed. by Fred Long. New York: Springer-Verlag, 1989.

[47] Young, Douglas A. *X Window Systems Programming and Applications with Xt*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[48] Zelkowitz, Marvin V. "Use of an Environment Classification Model" ACM/IEEE 15th International Conference on Software Engineering. Baltimore, MD, May 1993 (to appear).

Appendix A: Service Definitions from the *Reference Model for Frameworks of Software Engineering Environments (NIST SP 500-211)*

The numbers in parenthesis are section numbers as found in the Framework RM [6].

[These are the same as the service definitions in the frameworks section of the *Project Support Environment Reference Model (NIST SP 500-213)*.]

Object Management Services (4) - The object management services support the definition, storage, maintenance, management, and accessing of data objects and the relationships among those data objects.

The *Metadata Service (4.1)* defines, controls, and maintains metadata, typically according to a supported data model.

The *Data Storage Service (4.2)* defines, controls, and maintains objects, typically according to defined schemas and type definitions.

The *Relationship Service (4.3)* defines and maintains relationships among OMS objects.

The *Name Service (4.4)* supports naming objects and associated data and maintains relationships between surrogates and names.

The *Distribution and Location Service (4.5)* manages and accesses distributed objects

The *Data Transaction Service (4.6)* defines and enacts transactions.

The *Concurrency Service (4.7)* ensures reliable concurrent access to the OMS.

The *Operating System (OS) Process Support Service (4.8)* provides the ability to define OS processes (i.e., active objects) and access them using the same mechanisms used for objects, i.e., integration of process and object management.

The *Archive Service (4.9)* allows on-line information to be transferred to off-line media and vice-versa.

The *Backup Service (4.10)* restores the development environment to a consistent state after any media failure.

The *Derivation Service (4.11)* supports definition and enactment of derivation rules among objects, relationships or values (e.g., computed attributes, derived objects).

The *Replication and Synchronization Service (4.12)* provides for the explicit replication of objects in a distributed environment and the management of the consistency of redundant copies.

The *Access Control and Security Service (4.13)* defines and enforces access control rules on the OMS.

The *Function Attachment Service (4.14)* attaches or relates of functions or operations to object types or object instances.

The *Common Schema Service (4.15)* provides a means to create common (logical) definitions of the objects (and operations) from the underlying objects in the OMS.

The *Version Service (4.16)* manages data from earlier states of objects in the OMS.

The *Composite Object Service (4.17)* creates, manages, accesses, and deletes composite objects, (i.e., objects composed of other objects).

The *Query Service (4.18)* is an extension to the data storage service's "read" operation. It provides capabilities to retrieve sets of objects according to defined properties and values.

The *State Monitoring and Triggering Service (4.19)* enables the specification and enaction of database states, state transformations, and actions to be taken should these states occur or persist.

The *Data Subsetting Service (4.20)* enables the definition, access, and manipulation of a subset of the object management model (e.g., types, relationship types, operations if any) or related instances (e.g., actual objects).

The *Data Interchange Service (4.21)* offers two-way translation between data repositories.

Process Management Services (5) - This service group supports the unambiguous definition and execution of software development process activities across software life cycles.

The *Process Development Service (5.1)* provides for the creation, control and maintenance of process definitions.

The *Process Enactment Service (5.2)* provides for the instantiation and execution of process definitions by process agents that may be humans or machines. It also provides services to access, maintain and control the persistent state of the process.

The *Process Visibility Service (5.3)* provides facilities for the definition and maintenance of visibility and scoping information.

The *Process Monitoring Service (5.4)* observes the evolving enactment state of processes, detects the occurrence of specific process events, and enacts other processes to response to detected events.

The *Process Transaction Service (5.5)* supports the definition and enactment of process transactions.

The *Process Resource Service (5.6)* provides a means of assigning process agents to enact various processes and process elements.

Communication Service (6) - This service group supports inter-tool and inter-component communication among the tools and components of the SEE.

The *Data Sharing Service (6.1)* provides data sharing operations within the OMS or memory or by other data manipulation services.

The *Interprocess Communication Service (6.2)* provides primitive operating system process communication via the RPC mechanism of the network service.

The *Network Service (6.3)* supports communication among collections of processes and transfer of data between machines.

The *Message Service (6.4)* supports managed communication among a large number of elements of a populated environment framework.

The *Event Notification Service (6.5)* supports the notification of messages based upon certain triggering conditions.

Operating System (OS) Services (7) - These are the low-level platform functions.

The *OS Process Management Service (7.1)* creates, manages and schedules OS processes.

The *OS Environment Service (7.2)* passes information among OS processes.

The *OS Synchronization Service (7.3)* synchronizes the execution of OS processes.

The *Generalized Input and Output Service (7.4)* provides basic operations to transfer data between processes and input and output devices attached to a PSE.

The *File Storage Service (7.5)* provides the basic operations to read and write files and to store and access them in directories.

The *Asynchronous Event Service (7.6)* creates and sends signals between OS processes.

The *Interval Timing Service (7.7)* sets and tests timers on individual OS processes.

The *Memory Management Service (7.8)* manages the main memory.

The *Physical Device Service (7.9)* manages physical devices.

The *OS Resource Management Service (7.10)* provides general computer system management.

User Interface (UI) Services (8) - The subject of user interfaces is an extremely complex issue which is far more general than integration frameworks. Nevertheless, a consistent User Interface Service may be adopted for a complete framework.

The *UI Metadata Service (8.1)* describes the objects used by the UI Services.

The *Session Service (8.2)* initiates and monitors a session between the user and the environment.

The *Text Input Service (8.3)* provides for textual input processing by application programs.

The *Dialog Service (8.4)* provides the interface between the application program and physical display devices.

The *Display Management Service (8.5)* provides for interaction among individual windows.

The *Presentation Service (8.6)* creates and manages the physical interface between the user and the environment.

The *User Interface Security Service (8.7)* provides the security constraints needed by the UI.

The *User Interface Name and Location Service (8.8)* manages multi-user and multi-platform environments permitting communication among sessions, tools and display devices.

The *Internationalization Service (8.9)* provides capabilities concerned with different national interests.

The *User Assistance Service (8.10)* provides a consistent feedback from various tools to the user for help and error reporting.

Policy Enforcement Services (9) - These services provide the functionality of security enforcement and integrity monitoring.

The *Security Information Service (9.1)* supports the establishment of security information for use within the PSE.

The *Identification and Authentication (9.2)* provides for the ability to identify users and to properly associate them with appropriate access rights prior to any operations being carried out on their behalf.

The *Mandatory Access Control Service (9.3)* provides capabilities to assign access values by a security officer to govern access to the information contained in an object.

The *Discretionary Access Control Service (9.4)* provides the ability to permit users to control individual modes of access to objects that they own by individual users and all members of sets of users.

The *Mandatory Integrity Service (9.5)* provides the capabilities to protect objects from unauthorized or unconstrained modification as determined by the PSE security officer.

The *Discretionary Integrity Service (9.6)* provides the capabilities to protect objects from unauthorized or unconstrained modification as determined by a user.

The *Secure Exportation and Importation of Objects Service (9.7)* provides the ability to export and import objects in a secure manner.

The *Audit Service (9.8)* provides the ability to record information about calls on the PSE facilities in order to track and control security related actions.

Framework Administration and Configuration Services (10) - These services support the administration of the framework.

The *Registration Service (10.1)* incorporates new tools into an environment based on the framework effectively coordinating the new tool and the environment.

The *Resource Management Service (10.2)* manages, models and controls environment resources.

The *Metrication Service (10.3)* collects technical measurement information for the framework.

The *Sub-Environment Service (10.4)* provides the capability to divide the SEE into separate sub-environments and to restrict the access s of users to a subset of the available SEE resources.

The *Self-Configuration Management Service (10.5)* supports the existence of many simultaneous coresident configurations of a framework implementation.

The *License Management Service (10.6)* enforces of licensing requirements on components.