Rapid Response Manufacturing Consortium

Infrastructure Services Architecture

A Reference Architecture
for the
RRM Engineering Environments

Larry L. Johnson

Member, Group Technical Staff
Texas Instruments, Inc.

Abstract. An Infrastructure Services Architecture is presented for the Enterprise Integration Architecture framework of the Profile for Enterprise Integration. The architecture enables flexible re-engineering of the business process, and the construction of product-centered virtual enterprises. The architecture accommodates distributed, heterogeneous systems, providing protocols of service invocation and cooperation. Emphasis is placed on following the current trends and directions and industry at large, and on taking advantage of current and emerging commercial-off-the-shelf software.

TABLE OF CONTENTS

1. Introduction

1.1 Scope

This document describes an architectural framework for the implementation of an effective and evolvable engineering environment involving diverse applications. A framework of architectures is used to describe an infrastructure which accommodates multiple levels of abstraction ranging from the specific dimensions of the business process itself to highly generalized services of persistence.

1.2 Intended Audience

This document is targeted at technologists who will be involved in the specification, purchase, implementation, and/or integration of Engineering Environments for the Rapid Response Manufacturing Consortium (RRM).

1.3 Definition of Architecture

The word "architecture" means many things to many people. The use of the word in this paper simply connotes "the documentation of the parts of some whole and their inter-relationships." Usually there are many ways something can be viewed and inter-related so there is no single representation that can serve all purposes.

Taking the example of a building architecture, there are different representations required for the description of landscaping, framing, plumbing, electrical, etc. Generally, each of these representations will delineate some standard way of connecting parts within itself (standard electrical hardware, standard plumbing, or drawings of "typical" frame corners). Additionally, dependencies among one another are documented, showing the plumbing and electrical in the context of framing diagrams, for example.

Enterprises and information systems can be treated in much the same way, viewed through multiple perspectives of varying detail, each requiring its own representation(s) [9]. Various aspects of enterprise activities are examined from multiple points of view, and those points of view are made consistent among each other. We will examine a framework for the evolution of the business process itself, the architecture of an infrastructure to support the business process, and the inter-relationships between the business process and its implementing infrastructure.

1.4 Approach Strategy

The principal goals of implementing an Integration Services Architecture are to:

- Enable an ***evolutionary environment*** for the business process, facilitating expansion and change of the enterprise conceptual data model, and suite of application programs.
- Provide a mechanism through which diverse applications across ***diverse disciplines can share information*** without compromising its consistency.

- Provide a mechanism through which product information can be exchanged among a company and its ***customers and business partners***.
- ***Minimize independent entry and modification*** of dependent data without introducing excessive complexity.
- Provide a mechanism through which ***legacy applications*** can be integrated.
- ***Minimize in-house development*** of software and utilities.

In order to meet these goals, it is crucial that the infrastructure be compatible with those of other enterprises. Broad acceptance of an infrastructure architecture provides:

- ***Incentive for commercial application providers*** to write compatible and supporting applications, as well as broadly accepted exchange mechanisms among organizations.
- Widely accepted ***information exchange mechanisms***.

We therefore look to directions in the ***commercial marketplace and standards organizations***. A slavish approach to standards is not taken. Standards that can provide leverage are carefully incorporated in the architecture along with approaches to risk mitigation should one of the target standards fail to gain broad acceptance.

An Infrastructure Services Architecture will:

- Guide our purchases
- Help us make intelligent demands on our vendors
- Allow us, when necessary, to write software that will cooperate with other applications in the environment.

An important tenet of this document is that an Infrastructure Services Architecture must be ***defined within the context of the business process***. The ISA is not a collection of unattached, free-floating technological principles. Each aspect of the architecture must exist to support the business process itself. Consequently, we begin the definition of the Infrastructure Services Architecture within the context of an Enterprise Integration Architecture, which treats the business process as the top level of the infrastructure.

2. Enterprise Integration Architecture

2.1 Introduction

The problem of integrating applications that span the life-cycle of product data embraces many of the problems of general enterprise integration. Consequently, the infrastructure architecture is approached through a framework for enterprise integration that positions the architecture in the context of the business process and business process engineering.

2.2 Business Engineering Framework

The Enterprise Integration Architecture provides a framework for business process engineering. The framework consists of four zones of activity that define business domains and execute the business process for the enterprise within them (Fig. 1). The business process is then enabled and facilitated by an Infrastructure Services Architecture.

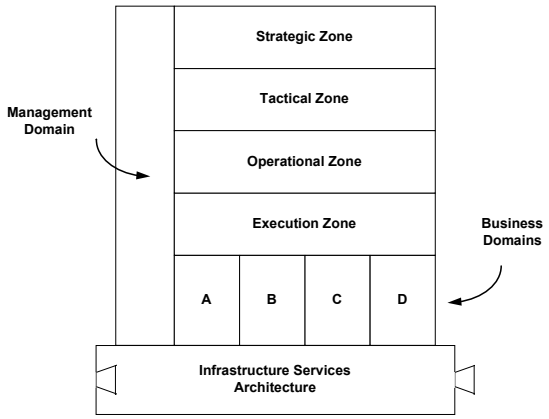

Figure 1. Enterprise Integration Architecture provides a business engineering framework for the continuous process of business engineering.

Strategic Zone: The activities that take place in the strategic zone define the vision, goals, and character of the enterprise. The business objectives, target market, broad personnel policies, etc., are defined and constitute the guiding principles and constraints for all activities in the enterprise.

Tactical Zone: The business domains required in order to achieve the enterprise goals and vision are delineated together with the interactions that are required among each. A typical corporation, for example, would require such things as Marketing, Sales, Engineering, Manufacturing, etc. Some corporations might further refine such broad categories as Engineering into System Engineering, Mechanical Engineering, Electrical Engineering, etc. The number of business domains, their scopes, constraints, and requirements are entirely determined by the business process engineers in order to optimize the enterprise toward the achievement of its objectives.

Operational Zone: The operational characteristics of each business domain are defined resulting in detailed Standard Operating Procedures and Guidelines. The business process of the domain is optimized within the constraints of its customers and vendors (other business domains).

Execution Zone: The business process is executed. The process is continuously monitored and metrics are collected to guide the continuous process of business engineering.

2.3 Management Business Domain

The management business domain is special in the sense that it is present from the start. All zones of activities are conducted within or involve the management domain. All other business domains are managed through this domain. The core requirement imposed on the management process is that it must:

- Define metrics to associate with each goal and objective in order to determine if the enterprise is on track and to identify any conflicting goals.

- Provide mechanisms for reviewing the metrics and changing the course of the enterprise based on the metrics.

Since the management mechanism operates across all zones of activities, policy management and changes will also take place across those activities. Changes will range from the redefinition of corporate goals to the introduction of a new manufacturing process to reduce costs.

2.4 Scalability of Enterprise Integration Architecture

Each of an enterprise's business domains is itself an enterprise. Its goals and visions are formed in the context of its containing enterprise (Fig. 2). As we will see in the discussion of Virtual Enterprises, a business domain will often find itself contained by many enterprises. This requires each business unit (enterprise) to optimize against all containing enterprises. Frequently, conflicts will be found due to cross-purposes of containing enterprises. When this happens, the conflict is fed back to the containing enterprises for coordination; their goals being brought into alignment.

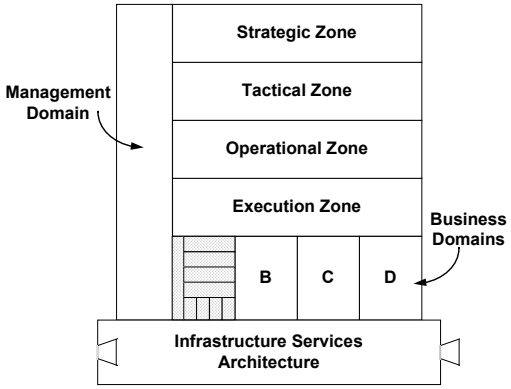

Figure 2. Business units operate as enterprises within enterprises.

2.5 Virtual Enterprises

The focus of the Enterprise Integration Architecture is the Virtual Enterprise. The Virtual Enterprise is centered on an enterprise product. Resources are drawn from the business domains in order to execute the business of producing that product. A product may be a single part or a strongly related family of parts (Figure 3). A virtual enterprise may span many physical corporations. The goals of the Virtual Enterprise must be harmonious with those of the physical enterprises from which the resources are drawn.

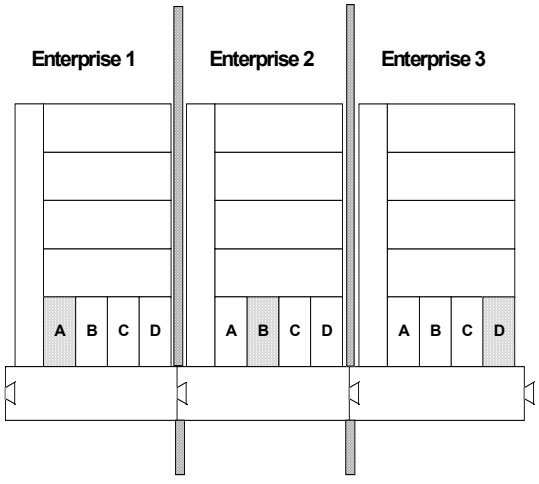

Figure 3. Virtual Enterprise defined across multiple enterprises (corporations or departments)

A corporation generally has a number of products or services it provides. The corporation provides disciplined pools of expertise through the establishment of it business domains. A virtual enterprise is established to produce each major product or family of products, drawing from that pool of expertise. The business domains operate as vendors to the product-centered virtual enterprise.

The context of the virtual enterprise is the ***product*** implying the inclusion of customers who need, demand, or require the product and vendors who

supply resources to produce the product. The vendors and customers are considered part of the virtual enterprise since each has a role in the production of the product. The case is easy to see for including vendors and subcontractors. Including the customer is a little more subtle, but recent advance in the culture of business dictates that the customer be part of the team, playing a key role in defining the product and its timely delivery.

Consequently, even a Sole Proprietorship with "no partners" participates as a member in multi-corporate enterprises.

2.6 Infrastructure Services Architecture

The business process of each virtual enterprise represents the top of the Infrastructure Services Architecture. Underneath the business process are services that enable an efficient and flexible execution of that process. The Enterprise Integration Architecture provides the context of the Infrastructure Services Architecture, keeping its mission focused and preventing it from becoming disembodied technology.

The rest of this document focuses on the definition of this architecture and its implementation.

3. Infrastructure Services Architecture Overview

In discussing an Infrastructure Services Architecture (ISA), architecture that exists in current de jure or de facto standards has been used wherever possible. However, the full body of any referenced standard should be consulted for that standard's architecture and position.

As described earlier, the ISA is depicted through a variety of architectural aspects. Each aspect has its own representation(s) to document it. Table 1 lists a set of architectural aspects and representations for each. These "architectures" are described later in this document.

Table 1. Architectural aspects of the Infrastructure Services Architecture and their Representations.

Architectural Aspect	*Architectural Representation*	*Benefits*
Business Process	• **Data Flow Diagram** - Documents the business functions, the data required to execute them, and the data they produce. Additionally, the sources of the data used and the users of the data produced are documented.	• Uniform results in conducting business • Facilitates re-engineering • Used in planning Application Architecture
Business Information Map	• **Table of Correlation** between Business Process Data Flows and Conceptual Schema.	• The conceptual schema provides a common point of reference for all business processes. • Helps prevent the "same" data being defined differently. • Helps prevent redundant data definition.
Conceptual Object Schema	• **EXPRESS Data Model** • **EXPRESS-G Diagram**	• The conceptual schema provides a common point of reference. • Basis for coordinating information

Table 1. Architectural aspects of the Infrastructure Services Architecture and their Representations.

Architectural Aspect	*Architectural Representation*	*Benefits*
		definitions among different business domains. • Basis for coordinating use of physical file formations and data locations.
	• **Object Method Map.** EXPRESS is not explicitly designed to capture object methods beyond those implied by the SDAI ("Get, Put style" methods, and data constraint validation). This provides an extension that documents the general methods of the objects in the EXPRESS model.	• Provides part of unification documentation between CORBA and SDAI. • Provides basis for constructing OMG IDL service descriptions. • Provides basis for mapping between formal object model and object service architecture. • Provides basis for generating API bindings.
Physical Information Map	• **Table of Correlation** Conceptual Schema and physical representations, formats and locations..	• Provides correlation between information consumed in a business process and the data format consumed by an application which is used to implement the process. • Helps track the translations necessary to support a particular application architecture in support of the business process. • Helps identify "dead-ends" wherein only one-way data translations exist (when traced back to the business process).
Application Architecture	• **Table of Correlation** between Business Process and Applications which implement the process	• Provides inventory of applications which can be used to execute a business process. • Provides support for selecting necessary application suite and translators to plan the implementation of a business process without dead-ends.
	• Application Source/Sink. Conceptual Schema/Format Table.	• Inventories the information required by and produced by an application. • Documents the physical form(s) required by the application for use of the information. • Provides support for selecting necessary application suite and translators to plan the implementation of a business process without dead-ends.
Object Service Architecture	• **API Documentation**	• Provides Documentation of the profile of services offered within the class as well as detailed documentation for the use of each service itself.
	• **Service Dependencies**	• Documents the inter-dependencies of services. • Assists developers in managing class boundaries and fan out.
	• **Application Service Register**	• Inventories the use of services by applications. • Assists developers in determining impact of service changes. • Determine service suites which must be

Table 1. Architectural aspects of the Infrastructure Services Architecture and their Representations.

Architectural Aspect	*Architectural Representation*	*Benefits*
		made available in an object service environment for operation of an application.
Domain Specific Architectures	• (See representations of the Object Service Architecture, above)	• Provides standard services to implement policies of business domains or functions (such as configuration management, CITIS services, design library support, etc.) • Services are closer to the business process than the low level core services of the Object Services Architecture. • Allows evolution of applications employing the services, permitting flexible support for re-engineered business processes.

3.1 The Business Process

In developing the Business Domain Architectures and their interactions, the business process is defined. The Infrastructure Services Architecture provides support for the execution of that business process.

The business process is executed by people with some degree of support from automation. The ISA is concerned with this automated support. The goal of having an ISA is to provide automation which:

- facilitates the efficient execution of the business process, allowing minimum cycle time in responding to market demands;
- can be reconfigured to support changes in the business process as required by our new world economic milieu.

In order to provide optimal support to the business process, the ISA must not exist wholly apart from business process; i.e., technology for technology's sake is not good business. It is the application of technology toward the accomplishment of business goals that provides the lift needed by business to reduce costs and provide quality products.

The technical details of the ISA are driven by, derived from, and traceable to the business process. Consequently, the top-level of the infrastructure is the business process defined in the Business Domain Architectures.

3.2 Application Architecture

The application architecture maps the business tasks of the corporation to support tools (Fig. 4). The information consumed and produced by a particular application as well as the required format of that information is documented. The goals of the corporation are achieved through the execution of the tasks defined in the business process. Each of these tasks can be mapped to a network of one or more subtasks that are:

- Strictly Manual,
- Strictly Automated, or
- Implemented through Manual use of an Automated Tool.

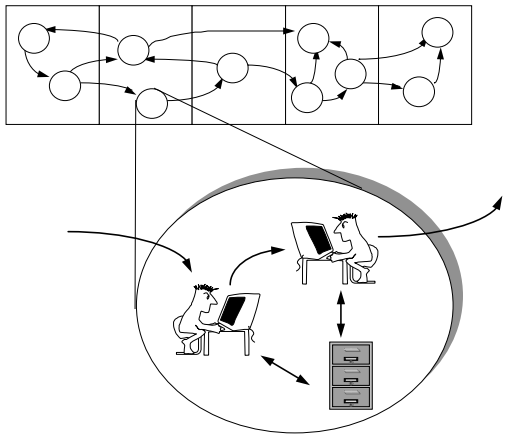

Figure 4. Application Architecture. The business process is mapped to supporting applications.

The detail provided in the documentation of the business process is kept as free as possible of assumptions of the use of specific applications. This allows the business process to enjoy the benefits of new application technologies as they emerge.

The application architecture dictates to some degree, what is possible in a business process. An inflexible application architecture can hold the business process hostage by making changes costly and intractable. When the business process is changed, it is often necessary to rearrange the order or concurrency of tasks. Monolithic applications resist change whereas those based on a services architecture (described later) lend themselves to being modified or rewritten.

In either case, the documentation of the applications and their relationship to the business process provides the information necessary to determine viability and/or costs of changes in the way of doing things. This is particularly valuable when supporting the business process through commercial software. This application architecture, in conjunction with the information architecture, helps identify the need for translators and other integration aids.

3.3 Information Architecture

Data is useful insofar as it can be interpreted as information. Applications interpret data through internal forms and constructs, but it is often stored in very different structures. Many applications will use the same information, but often, each application will require the information to be in a particular format (physical form) that is different from the format used by others. Furthermore different applications may interpret the same information from its particular point of view. However, conceptually this information must be the same in order to share it. We adapt the ANSI SPARC three-schema architecture to map the information among these different contexts. This mapping provides a foundation that will enable the construction of an information access strategy that, at some level, is independent of its physical form.

3.3.1 Conceptual Domain Model

Key to the ability to exchange information among enterprises is an agreement on the meaning of data being exchanged (semantics), and how the data are represented (format). Historically a great deal of attention was provided to the latter problem of format standardization, but as multi-corporate partnering became an imperative it was realized that the approach was wholly inadequate. It is necessary to agree on the meaning of the data as well as its form, or there is no basis for a meaningful exchange of information. Anytime a meaningful exchange of information takes place, there is an explicit, implicit, or serendipitous agreement on the conceptual schema, i.e., the fundamental meeting of the data.

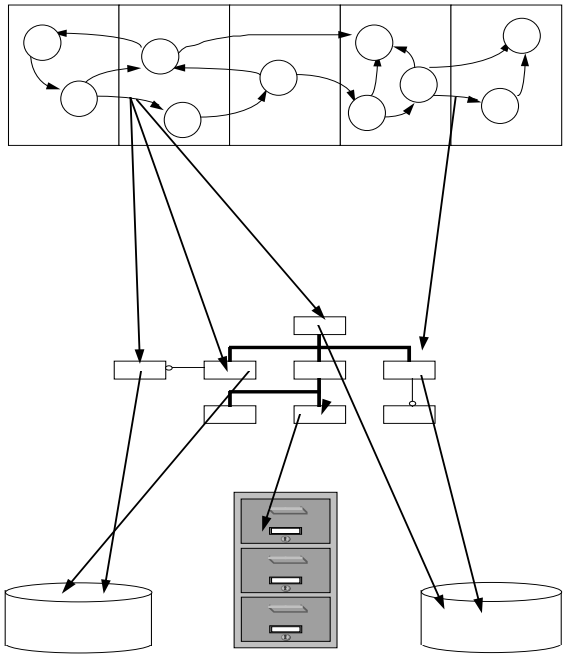

Figure 5. Information requirements of the business process are mapped to the conceptual schema that is then mapped to physical data representations.

The physical form or format (data) of information can vary widely. The necessity of agreement at the conceptual level makes the Conceptual Domain Model at the center of Information Architecture.

3.3.2 Physical Schema

The physical schema of information is the specification of its precise physical form as data. Information can be transmitted via a variety of physical media (e.g., paper, magnetic, network). In each of these media the information can be encoded in a variety of physical formats (e.g., HPGL, IGES, Postscript)

The physical schema and conceptual schema sometimes get "tangled" in some instances where a format specification limits the information that can be encoded. For example, only certain concepts can be represented in an IGES file whereas the STEP file format can be used to store data of an arbitrary conceptual schema. The same conceptual elements can appear in a variety of physical schemata. Consequently the mapping of the conceptual schema to physical schemata will in general be one-to-many.

3.3.3 Application Schema

A given application requires information in a specific format(s) that it understands. Consequently, it is important to establish a correlation between the information required in the business process, the applications implementing that process, and the available physical forms of that information. Using this information, a viable application architecture can be established.

3.3.4 Information Mapping

The data flows of the business process map to some aspect of a conceptual model (Fig. 5). To develop an application architecture to implement the processes, it is important to know what information is consumed and produced by the business process and then what form that information needs to assume in order to be consumable by the particular application. For example, a CAD system produces and alters (consumes) geometry. The geometry that is consumed must be in a suitable format to be consumable by any given CAD system. The information may be in a proprietary format, IGES, STEP, etc. If an application is to be used in the business process, it is important that the ramifications of format conversion (and any attendant information loss) be taken into account.

We have accounted for business process information needs and demands by mapping the information required to the conceptual schema. We have also accounted for the business tasks by mapping them to manual procedures and automated applications. Applications, however, do not use abstract conceptual information, but physical data. Consequently, we must map our conceptual schema to a physical schema. The physical schema accounts for data format, access mechanisms, and data locations.

3.4 Object Architecture

The term "object model" is used in the literature in two ways:

1. An **Object Schema**, i.e., a description of objects, their attributes, and behavior. This is the way the term will be used in this document.
2. An **Object Meta-Model**, i.e., a description of the semantic constructs supported by a modeling representation or framework. In this document the term "meta-model" will always be used to avoid confusion.

3.4.1 Object Paradigm

The business process model with its attendant application architecture provides a process view of the system. Traditionally, process oriented applications were written which did things to data objects. As E/R Modeling progressed, the data objects took on a one to one correspondence with the "things" of the problem space. What object orientation does is to associate the action with the object itself, as part of the object... the action is said to be one of its methods. The method is invoked by sending a message, or request, to the object itself. Object applications execute by sending messages to accomplish their desired end. The information model taken together with the action (method) model constitutes an object model. The object model serves as a conceptual domain model.

The information aspect of an object model provides the "nouns" and "adjectives" of an object vocabulary. The operational aspect of the object model (the methods) provide the "verbs" of an object vocabulary. What is required is a "sentence syntax" on which applications can agree. The object vocabularies together with a syntax for using that vocabulary constitute an object execution language.

An operational system requires the addition of a communication medium. Since many media may be used, we require isolation from it, so that our language is not dependent on the media (we do not want to be aware of the media at all).

The syntax and communication mechanism are provided by an "object request broker" (ORB). Given a message, the ORB determines the target object(s) and delivers the message to it for execution.

The specification of the form(s) and details of the object model and the ORB constitute the Object Architecture.

3.4.2 Object Model

In order to facilitate present and future integration, it is important that data to be exchanged be agreed upon through the information architecture. Within the PDM domain, there must be agreement on a "master" conceptual schema for information that is to be exchanged..

The conceptual domain model of PDM needs to be represented so that it is compatible with the rest of the object architecture. Conventions that can capture both the informational and behavioral aspects of objects must be adopted. A synthesis of approaches from current industry directions is suggested later in this document (STEP conceptual domain model, and the OMG Object Services Architecture).

4. Object Services Architecture

Figure **6** shows the Object Services Architecture (OSA) taken from the Object Management Group (OMG) [1], [2], [3]. Among the desirable features of the architecture are:

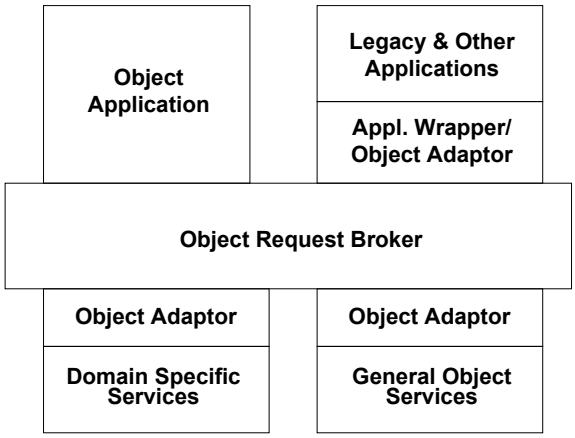

Figure 6. Object Services Architecture.

- Removal of a "system" from the center. Many of the early approaches to integration involved some omniscient central system. The OSA has at its center a convention of semantic communication protocols. No hardware. No software. Simply semantics and a protocol for "speaking" in terms of the semantics.
- Sparseness of core services. Primarily the architecture is a set of services minimally necessary to add and invoke sets of services required by particular application domains, or by data objects in general.
- Location transparency. Requests for action are not necessarily sent to a specific location. In fact, there may be no way of knowing (or need to know) where the desired action can or will take place.

4.1 Object Request Broker

One of the key aspects of an object system is its messaging capability, which separates the action from the requesting application. To make use of legacy applications we need to be able to make them respond to object messages, isolating their implementation details (language, location of service, operating system, hardware, etc.). This shifts the "center" of the system from hardware and software to a protocol of invocation and communication (Fig. 13). The legacy applications shown making a request and responding to it in Figure 13, can be on any operating system, at any place, and written in any language, providing that an Object Request Broker protocol is used to standardize the service request and its response. The movement of the request and the response is the responsibility of the request broker.

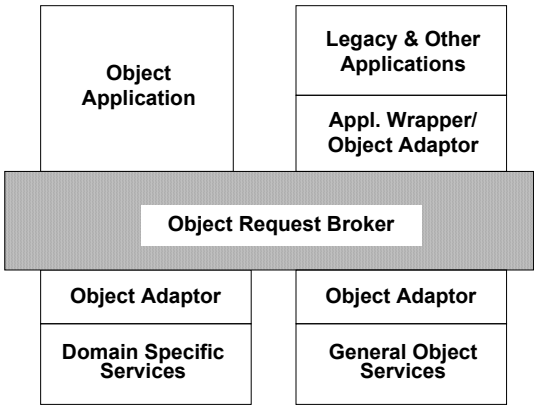

Figure 7. Object Services Architecture. The Object Request Broker forms the backbone of service requests.

4.1.1 Message Transport

In object-oriented activities, messages are sent to objects, requesting them to perform some action. The request broker provides the convention on how to form or understand an object request (sentence) providing for specification of the object (subject), the operation (verb), and parameters (modifiers and clauses). Messages are created and "parsed" through standard messaging services, hiding the details of the messages, packets, etc. The object request broker (Fig. 7) provides the common language syntax and conversation protocols of the system. (The actual vocabulary of the system will be provided by domain specific object models provided through the semantic services of the infrastructure.)

The ORB uses a Client Stub to bind a service request to an application program (an Application Programming Interface, API, binding). CORBA supports early binding through the creation of Client Stubs for the particular language being bound. This binds the service to the application at compile time. CORBA also supports a late (run-time) binding. By using the IDL description of the object and service being requested, CORBA invokes the desired service. Neither the identities of the object, its type, nor the desired service need to be known by the application program until the request is made through CORBA's dynamic interface.

4.1.2 Schema Mapping

In 1974 a three-schema architecture was put forth as the ANSI/SPARC Database Architecture consisting of

1) Conceptual Schema - The implementation independent view of information from a conceptual viewpoint.

2) External Schema (or Application Schema) -- The schema of the data from the point of view of some particular application, program, or presentation.

3) Internal Schema (or Physical Schema) -- The physical organization of the information in an information storage (database) scheme.

A shared conceptual schema is essential for the sharing of data. If agreement cannot be reached on what the data are or what the data represent, then there is no hope of sharing the information. Whether an enterprise integration strategy is based on a master schema approach or a federated system approach, all information sharing is predicated on agreement at some point as to what the data represent conceptually.

If we treat our persistent storage (database) facility as just another application, then the Internal Schemata and External Schemata become private schemata (i.e., private to the application). This leaves the Conceptual Schema as the public schema.

The public (conceptual) schema provides the vocabulary of object requests, since

- private schemata will have a great deal of diversity, and
- information sharing is predicated on agreement on the conceptual schema

Applications therefore make requests of objects in terms of the conceptual schema. These requests are instantiated as Application Programming Interface (API) calls through an early binding mechanisms, or through the specification of an object, method, and parameters in a late binding, dynamic run-time request mechanism.

4.2 Object Adapters

The primary mechanism by which object services are provided by the ORB is through Object Adapters (Fig. 8) which are responsible for:

- Generation and Interpretation of Object References
- Method Invocation
- Interaction Security
- Object and Implementation Activation and Deactivation
- Mapping Object References to Implementations
- Registration of Implementations

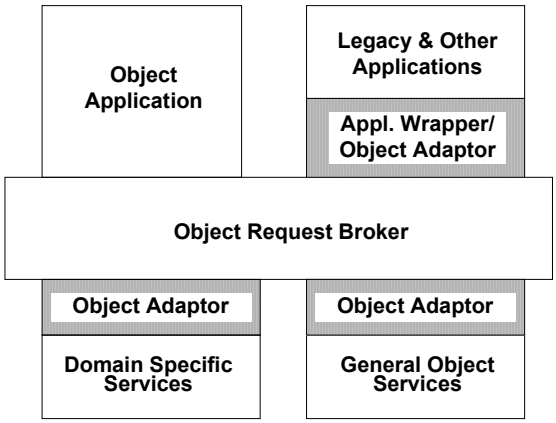

Figure 8. Object Adapters provide a uniform protocol of service invocation and management.

4.3 General Object Services

The intent of an object services architecture is to provide partitions of services that can be supplied by various vendors. This would allow a consumer to select Best in Class services from

multiple vendors. Similarly, it would allow these standard service partitions to evolve in a backwardly compatible fashion to new or expanded service partitions.

Intrinsically, the service partitions constitute the operations of an abstract object class, i.e., the methods of an object class that is independent of a particular application domain (Figure 9). In the definition of particular object domains, object classes are declared subclasses of applicable abstract object classes in order to impart the services and characteristics of the abstract classes. For example, Naming Services are provided for objects that are in the class of Named Objects.

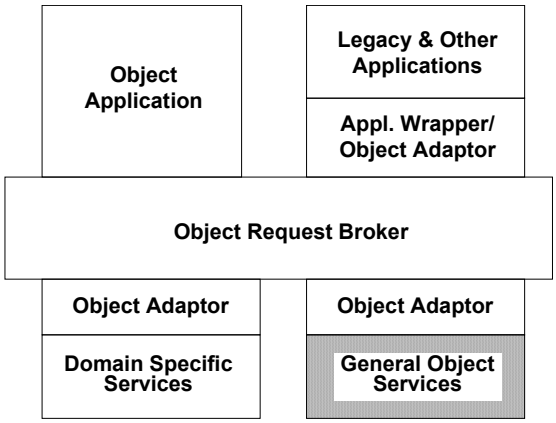

Figure 9. General Object Services. Fundamental object services that constitute a relatively abstracted view of objects.

Request for Proposals for service definitions have been adopted for [6]:

- Naming
- Events
- Life Cycle (create, delete, move)
- Persistence (object database storage)

Services that are expected to be adopted in the 1994 time frame are [6]:

- Relationships
- Time
- Externalization (Import Export)
- Transaction
- Concurrency

Services that are candidates (tentative) for adoption in late 1994 to 1995 are [6]:

- Security
- Trading
- Properties

4.3.1 Interface Repository

The interface repository holds definitions of the signature of a persistent object request. The signature is an implementation neutral description of the objects as well as its services and service parameters. The signature is recorded in Interface Description Language (IDL) which is defined as part of CORBA. The signature information is used by the ORB to perform requests. Beyond direct support of the ORB, the IDL signatures allow applications to operate on objects that were not known at compile time.

4.3.2 Implementation Repository

The services of the Implementation Repository are used by the ORB to locate an object implementation and invoke its services. To accommodate this the Implementation Repository contains the information required for the ORB to locate and activate object implementations. Additionally, the Information Repository is used to store debugging information, administrative control, resource allocation, security, etc.; i.e., anything that supports the dispatch of an object message and the invocation of an object service.

4.3.3 Persistence Services

The definition of a service suite for persistence has recently been completed by the Object Data Management Group, a consortium of database vendors and object technology companies including SunSoft, Object Design, Ontos, O_2 Technology, Versant, Objectivity, Hewlett-Packard, Poet Software, Itasca, Intellitic, Digital Equipment Corp., Servio, and Texas Instruments.

Each of the database vendors has committed to the provision of services compliant with the standard for their database implementation. The approach is a "Persistent C++" approach.

4.3.4 Semantic Services

Semantic services provide the mechanisms through which the vocabulary of the system is provided. The mechanisms provide means for determining what is a meaningful object request (or "sentence"), and what makes sense in a given domain of interest.

The semantic services are responsible for managing an object schema, providing the identification of object types, their attributes, relationships, constraints and available operations. The services must include the repository of operation signatures, constraint processor for ensuring the integrity of object state and relationships among objects.

Semantic object domains may be queried through the semantic services for information on which a dynamic object request might be based, but for the most part the services would be employed transparently to ensure data integrity.

Services are necessary to allow the declaration of overlapping semantic domains, and enabling the enforcement of constraints that may be unknown to a requester who is operating within a single particular semantic context.

Groups promoting semantic services are currently in a state of flux, but appear to be coming together. The ANSI X3H4 Information Repository Dictionary System (IRDS), ECMA's Portable Common Tool Exchange Environment (PCTE), and the Interface Repository of the Object Service Architecture of OMG seem to be converging and cooperating. In particular, the X3H4 committee has formally proposed its realignment from the traditional IRDS standard to the emerging international effort on PCTE. [4]

4.4 Domain Specific Services

The functionality embedded in applications often has to be replicated when it is required by other applications. When it is replicated, it is often replicated imperfectly. Furthermore, when the functions are changed or upgraded, the change does not occur uniformly, often resulting in applications that are incompatible for a period of time. Much of the functionality currently embodied in applications can be moved to higher order Domain Specific Services (see discussion of the PDM Domain Services on Page 23). These services implement policy and operational functionality in one place, allowing the functionality to be employed flexibly as the business process or its implementing applications evolve.

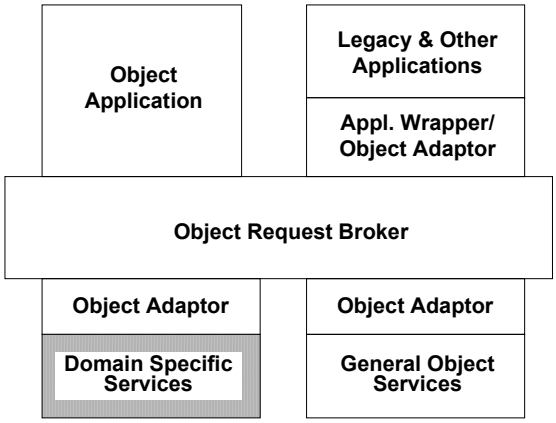

Figure 10. Domain Specific Services. Support of larger granularity objects and methods in the context of a business domain.

Applications then become primarily service requesters, the bulk of the work being done in domain specific services (Fig. 10). Environments under transition can have a mix of traditional applications that execute the policies and functions themselves, and newer object-oriented applications that do most of the substantial work through API bindings to domain specific service requests.

4.5 Functional Architecture (Service Dependencies)

The Object Service Architecture (OSA) provides a service invocation architecture and the ability to accommodate numerous classes of services. The OSA does not explicitly provide the detailed functional dependencies or layered view that is often seen in architectural documents, e.g., the Open Systems Interconnect (OSI) layered network architecture. However the OSA does suggest an architecture of functional dependencies at an abstract level (Fig. 11). The specific concrete operation of executing the business process uses object applications which in turn use object services. A specific operation or set of operations from a particular business domain will often have domain specific services, which can be used which in turn employ other object services.

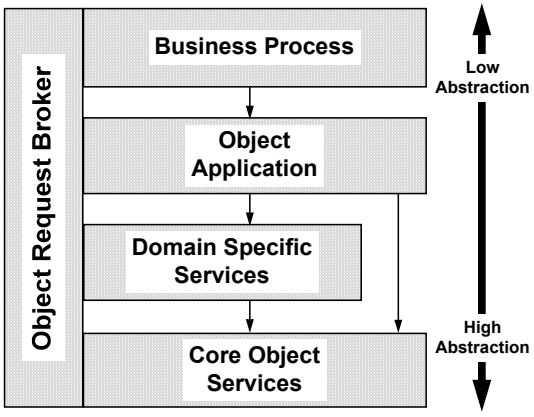

Figure 11. The Abstract Functional Dependency Architecture suggested by the Object Services Architecture. The more specific, less abstract services make use of the general, more abstract services.

This usage of service packages creates dependencies that must be tracked. As classes of Domain Specific Services emerge in number,

and approach the Object Applications themselves in granularity, the interdependencies will become numerous and complex. The replacement, supercession, or elimination of a service class can cause service disruption for those services dependent upon it. Consequently, service dependency databases must be maintained to aid in impact analysis. Such a database could be implemented as an adjunct to the OMG Service Repository.

4.6 Object Application Environment

Under the OMG Object Service Architecture, application programs request General, Domain Specific, and Legacy Services through the Object Request Broker in terms of the semantics of the conceptual object model (Fig. 12).

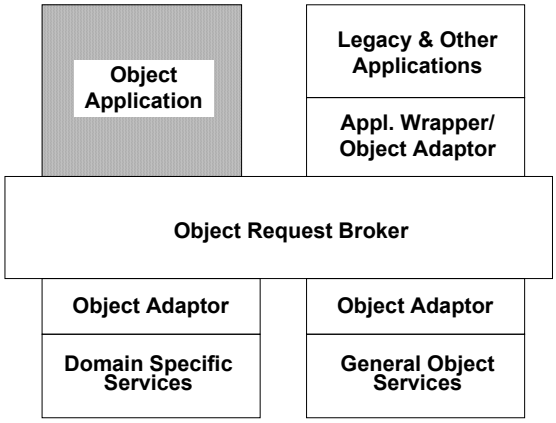

Figure 12. Object Application Environment. The object service architecture isolates the application from implementation and behavioral details.

Applications enjoy the following advantages by using the OSA:

- *Isolation* from the language in which the service was written
- *Isolation* from the operating system in which the service runs
- *Isolation* from the type of hardware hosting the service
- *Isolation* from the location of service provision.

Obviously, one of the key benefit of the OSA to applications is *isolation*. This allows service provision to enjoy a great deal of flexibility, allowing migration across hardware, operating systems and languages.

There are other benefits that accrue to applications as well. When providing object services through the OSA, one assures uniform behavior of the object to which the service is associated. All applications that wish to implement a particular behavior do not implement it themselves, but provide it through common service calls. Consequently, the risk of undesirable differences occurring in the behavior of an object from one application to another is eliminated. This has further ramifications when it is necessary to change the behavior of an object. When a change in behavior (service) is required, it need be changed only in the service provider, protecting the many applications that employ it from change.

4.7 Legacy Applications

The advantages of object-oriented technology are widely publicized, but the technology has not reached mainstream application (though that is rapidly changing). Part of the impediment to the switch to object technology is the massive legacy of applications that exists. An architecture that permits the evolution rather than wholesale overthrow of our existing systems is needed.

Consequently, an important requirement on the ISA is the ability to accommodate legacy applications. The object service architecture supports this through "wrappers" that are special object adapters (Fig. 14). The object adapter understands the requests made by the Object Request Broker at one "end" and application specific code "uses" the Legacy Application through a "shell", or through an older style Application Programming Interface. Responses are intercepted from the Legacy Application and returned to the requester in a CORBA compliant manner.

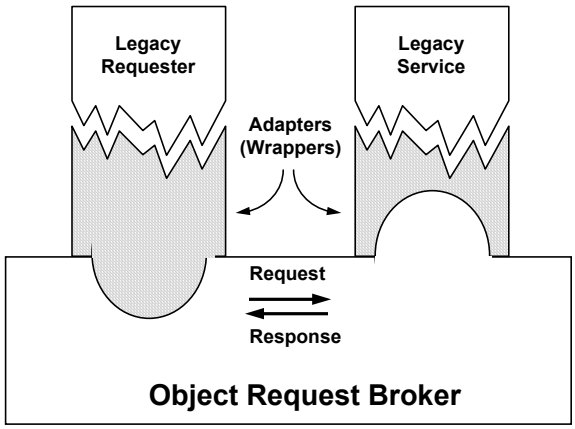

Figure 13. An object architecture that accommodates legacy applications. Adapters (also called wrappers) are written to provide a compliant interface to a uniform messaging protocol.

Adapters will be "thick" (more software will be involved) for legacy applications that were designed without regard to object technology. Adapters will be "thinner" for services that were written specifically for the architecture.

The limitation of this wrapper approach is dictated by the architecture of the application itself. It may be difficult or impossible to gain control of specific services of applications that were not designed for it. Even crude programs, however, can often be accessed through a "shell" that performs terminal emulation.

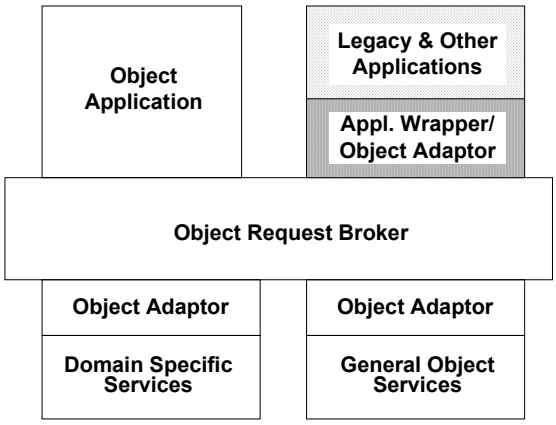

Figure 14. Object Adapters provide the "wrappers" through which services of Legacy Applications can be made available to the Object Applications.

5. STEP, the Conceptual Domain of Product Data

5.1 Introduction to STEP (ISO 10303)

Early work on data format for the representation of product data was done as IGES (the **I**nitial **G**raphic **E**xchange **S**tandard). It became clear that it was necessary to exchange more than mere graphics. To capture product data, the meaning underlying the data (the semantics) needed to be exchanged as well. As IGES evolved, it became clear that another approach was going to be necessary. The US formed the **P**roduct **D**ata **E**xchange **S**tandard (PDES) in order to expand the scope of the product data exchange effort to include the semantics.

Similarly, the **I**nternational **S**tandards **O**rganization (ISO) undertook the development of a standard conceptual schema for the representation of Product Data throughout its life cycle. The undertaking is being standardized as ISO 10303, and is called STEP (**ST**andard for the **E**xchange of **P**roduct data). The US effort merged with STEP and changed the meaning of the PDES

acronym to "**P**roduct **D**ata **E**xchange using **S**tep". The official US representative to the STEP effort is the **I**GES/**P**DES **O**rganization (IPO).

The methodology of STEP involves a two-layer semantic model. The **A**pplication **P**rotocols (AP's, Parts 200+) provide the context of the model. Each AP has its information mapped to the Generic Resource Models (Parts 40-99). The AP is activity centered, bringing together information that is involved in a particular life cycle of broad product families, sometimes particular to a particular industry.

STEP is organized into "Parts" whose numbers are grouped in topical ranges:

- Parts 1-10 Overviews
- Parts 11-19 Descriptive Methods
- Parts 21-29 Implementation Methods
- Parts 31-39 Conformance Testing and Methodology Framework
- Parts 41-99 Generic Resources
- Parts 101-199 Application Resources
- Parts 201-1199 Application Protocols.

5.2 Construction Tools of STEP

Parts 1 through 49 provide the tools and procedures for the construction and use of STEP semantic models. They cover representation, access, and conformance standards. The provision for explicit conformance specifications is notable. It is the result of lessons learned from IGES, wherein it has been found that any two applications that claim to be IGES conformant can rarely exchange IGES part descriptions with complete success.

In this category STEP currently defines[1]:

◊ ***Part 1.*** Overview and Fundamental Principles

◊ ***Part 11.*** EXPRESS Language Reference Manual -- Express is an object-oriented data modeling language that is used to describe the information domains of Parts 41 and above. The most distinctive feature of EXPRESS is that it is first and foremost a computer parseable language.

◊ ***Part 21.*** Clear Text Encoded Files -- Describes a format for encoding data that are compliant with a particular EXPRESS semantic model. These files are essentially the "son of IGES" exchange format capabilities, except that it accommodates the semantic context of an EXPRESS model.

[1]A hollow diamond bullet, "◊", denotes parts which were included in the **D**raft **I**nternational **S**tandard (DIS) version of STEP, 1993. The unmarked parts are in progress or soon to be started.

- ***Part 22.*** SDAI (Standard Data Access Interface) -- Describes a standard for the construction of an API (Application Programming Interface) based on a particular EXPRESS semantic model.

◊ ***Part 31.*** Conformance Testing Methodology and Framework: General Concepts

- ***Part 32.*** Test Laboratory Requirements

- ***Part 33.*** Structure and Use of Abstract Test Suites.

- ***Part 34.*** Abstract Test Methods.

5.3 EXPRESS Modeling Language

One of the foundation construction tools of STEP is the EXPRESS modeling language defined in Part 11 (ISO 10303-11). Express was developed to document the data models in an object-oriented fashion. Although it does not accommodate the general modeling of object methods that is required for full object modeling, it does provide for the modeling of constraints. EXPRESS is a textual documentation of an information model that follows strict syntax, making it easily parsable through automated means.

A subset of EXPRESS was given a graphical representation in EXPRESS-G to make it accessible to people in the way that NIAM and IDEF1X are. (Missing from EXPRESS-G is the accommodation of constraint definitions)

By focusing on the "computer digestibility" of EXPRESS, the STEP community enabled the development of "engines" or "toolkits" that are driven by EXPRESS, such as schema editors, automated API binding generation, dynamic data access, etc.[2]

EXPRESS distinguishes itself as a modeling representation by:

- Computer Processable Textual Representation Syntax
- Human Interpretable Graphical Representation
- Commercial Off-the-Shelf Software Support exists
- A Large Repertoire of Public Domain Models (Part Structure, Geometry, etc.)

Steadily Increasing International Acceptance inside and outside of the STEP community

5.4 Integrated Resource Models

Integrated resource models (represented in EXPRESS) are used by Application Protocols to describe the information of the AP's application context. The resource models are divided into two categories: the Generic Resource Models that describe information that is wholly

[2]Some of these have already become commercially available through such companies as STEPTools, Inc., and International TechneGroup Incorporated (ITI), Digital Equipment Corporation (DEC), and others.

independent of any particular application context, and Application Resource Models that describe information that is dependent on the application context.

The following Generic Resource Models (Parts 41-99) have been defined[1]:

- ◊ ***Part 41.*** Fundamentals of Product Description and Support.
- ◊ ***Part 42.*** Geometric and Topological Representation
- ◊ ***Part 43.*** Representation Structures
- ◊ ***Part 44.*** Product Structure Configuration
- ◊ ***Part 45.*** Materials
- ◊ ***Part 46.*** Visual Representation
- • ***Part 47.*** Shape Tolerances
- • ***Part 48.*** Form Features

The following Application Resource Models have been defined[1]:

- ◊ ***Part 101.*** Drafting
- • ***Part 102.*** Ship Structures
- • ***Part 104.*** Finite Element Analysis
- • ***Part 105.*** Kinematics

5.5 Application Protocols

The **A**pplication **P**rotocol (AP) is the unit of compliance in STEP. To be compliant with STEP is to be compliant with an application protocol.

The unifying principle underlying an AP is the Application Context which establishes the intended use of the data in some phase(s) of a business process.

An AP consists of an **A**pplication **A**ctivity **M**odel (AAM), **A**pplication **R**eference **M**odel (ARM), and **A**pplication **I**nterpreted **M**odel (AIM). The AAM and the ARM establish the Application Context (domain of applicability) for the AP. The AAM is typically provided as an IDEF0[3] model. The AAM is a non-normative portion of an AP, i.e., it is not involved in the measure of compliance to an AP. The ARM is an EXPRESS, NIAM[4], or IDEF1X[5] model[6] of the information employed by the AP (explicitly or implicitly through the AAM). The ARM and the AAM together establish the scope of activities and information that the AP is intended to address.

[3]IDEF0 is a graphical process modeling representation which is part of the US Air Force defined suite of IDEF tools.
[4]NIAM is a graphical information modeling representation which is popular in Europe and enjoys some degree of use in the USA.
[5]IDEF1X is a graphical information modeling representation which is part of the US Air Force defined suite of IDEF tools.
[6]There is increasing tendency and pressure to represent the ARM in EXPRESS. The accommodation of NIAM and IDEF1X was an early "treaty" to keep things moving in STEP.

The ARM is mapped to the integrated resource models (Part's 40-199) through the AIM. Since all AP's are constructed through mappings to the generic resource models, it is assumed that all AP's will be compatible among themselves. (Part of the validation process for AP's involves a review to ensure the compatibility of the AP with others.)

The following application protocols have been defined. Only Explicit Draughting (201) and Configuration Controlled Design (203) have been included in the Draft International Standard. The remaining AP's are in varying degrees of definition and completion.[1]

- ◊ ***Part 201.*** Explicit Draughting (2D)
- • ***Part 202.*** Associative Draughting (3D)
- ◊ ***Part 203.*** Configuration Controlled Design (Part Structure and Configuration Management).
- • ***Part 204.*** Mechanical Design using Boundary Representation
- • ***Part 205.*** Mechanical Design using Surface Representation.
- • ***Part 206.*** Mechanical Design using Wireframe Representation.
- • ***Part 207.*** Sheetmetal Dies and Blocks
- • ***Part 208.*** Life Cycle Product Change Process
- • ***Part 209.*** Design through Analysis of Composite and Metallic Structure
- • ***Part 210.*** Electronic Printed Circuit Assembly, Design, and Manufacture.
- • ***Part 211.*** Electronics Test Diagnostics and Remanufacture.
- • ***Part 212.*** Electromechanical Parts

6. Product Data Domain Services Architecture

The Product Data Management Domain is large and complex covering services that span the life-cycle. Consequently, the PDM Domain would not be a single service class, but would be subdivided into a number of subdomains covering design, configuration management, etc. The usage of services among the service classes would be carefully documented. Figure 15 shows possible service classes for the PDM domain that involve product design, configuration management, and CALS data management through Contractor Integrated Technical Information Services (CITIS).

The service classes and their rule bases would be specified by those responsible for the particular domain of activity. For example, the policy of configuration management would be implemented in one or more classes of service that are guided by one or more sets of rules maintained through a configuration management team or organization. When a configuration

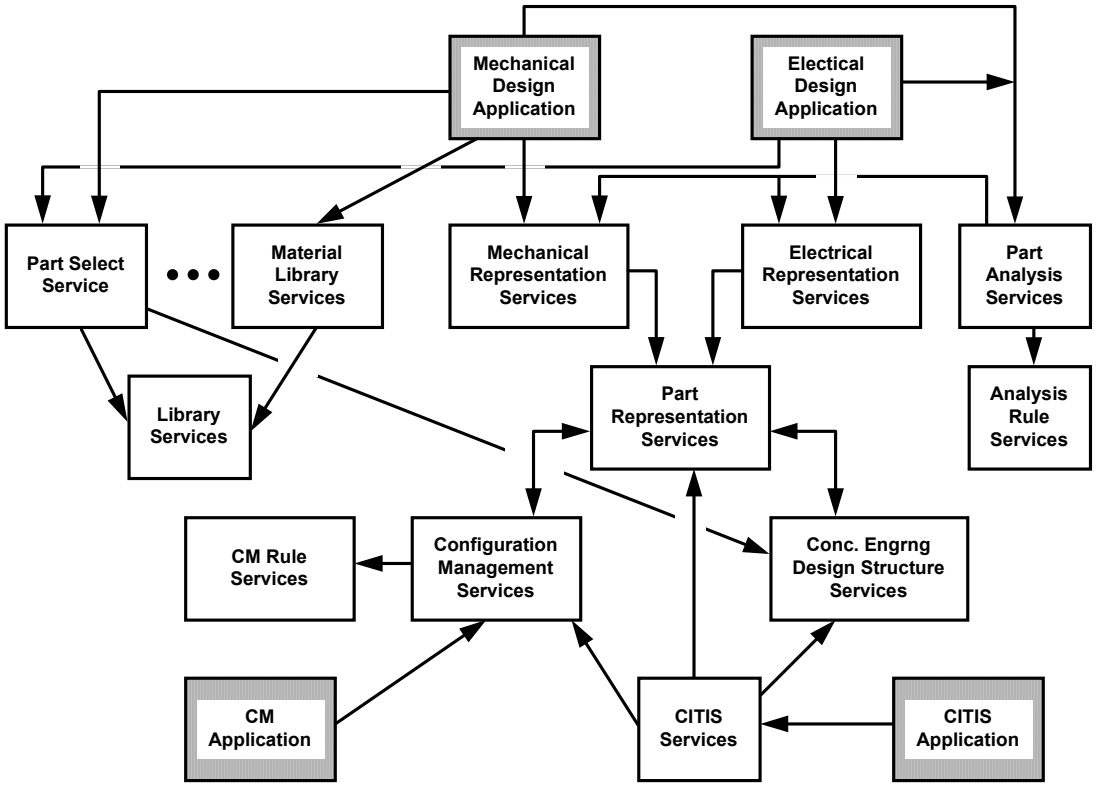

Figure 15 An example of a possible PDM Domain Services architecture. The directions of the arrows indicate usage of one service class by another. The service classes of the PDM domain cover the services required for managing the product and its design over its life-cycle. Applications that use the domain services are shown in shaded boxes. The use of Core Object Services by these Domain Services is not shown. (See Fig. 11).

management service or application of policy is needed, then the required service is called. This allows a great deal of cross-domain activity within a single application or function when necessary. Consequently, a designer can avoid incorporating a part that is not at an appropriate state of acceptance.

The essential point is that the services and rules are captured and maintained by the functional groups that have the particular expertise that is necessary, but the execution of those services can take place by anyone, anywhere in the business process that it makes sense.

7. Suggested Foci for RRM Architecture Activities

7.1 Business Process Mapped among Schemata

The highest level of the architecture aspects is the business process documentation, followed by its mapping to the conceptual schema (the IPPM). By mapping the target business process to the IPPM, a first order validation can be done on the IPPM. Information which is required but not in the IPPM, and information in the IPPM which is superfluous to the business process can be identified.

Additional benefits can accrue from further levels of mapping:

- By mapping the conceptual schema to the physical schemata, an accounting of the forms and formats supported in the engineering environment is provided.
- Mapping the business process to the applications provides an accounting of the process functions that are covered by particular applications or service profiles.
- Mapping the applications through the conceptual schema to the physical schemata which they require for input and which they produce, permits an audit to assure that there are no format mismatches from application to application. An inventory of the mismatches (disconnects) provides an inventory of the translators which are required.

These schema mappings provide validation of the business process, as well as coverage and representation of the information required.

7.2 Product Data Domain Services Architecture

The definition of domain specific services is one that may hold considerable opportunity for RRM. The services provided by current STEP tools and approaches are relatively low-level, providing for primitive create, update, and read of data elements. Services which are closer to the business process can be defined to enable the flexible design and redesign of applications and the business process that they support.

Possible service classes of interest to RRM are:

- Design Structure
- Design Analysis Functions
- Work Flow
- Configuration Management
- Library Services

7.3 Standards Activities

7.3.1 Harmonization of CORBA and SDAI

An architecture that brings together the Conceptual models of STEP and the Object Architecture of OMG is shown in Figure 16.

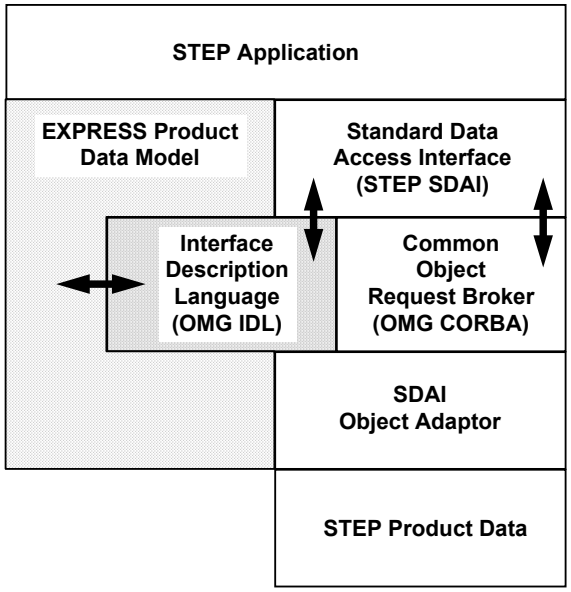

Figure 16. Architecture harmonizing the EXPRESS and Standard Data Access Interface (SDAI) of STEP with the Interface Description Language (IDL) Model and CORBA of OMG. The arrows indicate interfaces requiring significant harmonization work

The EXPRESS data modeling language of STEP goes beyond many modeling representations in being able to specify rich semantics of relationships and constraints among data object types. However, EXPRESS still falls short of an object modeling language because it does not support the specification of general methods for objects.

On the other hand, the IDL of OMG is primarily oriented toward the description of procedural service interfaces. It focuses on representing the signature (form and parameterization) of "procedure" calls. The IDL is based on an object model that supports only sparse semantics (characterized by multiple inheritance and a general abstraction of relationships).

The two standards brought together provide the conceptual elements necessary to put together a flexible object environment. There are difficulties, however. The SDAI and CORBA both have various explicit and implicit aspects of session models which must be made consistent.

Standards groups and industry consortia need to work on the harmonization of CORBA and SDAI. Work must be done in RRM to bring these elements together in a local, if not global, solution.

7.3.2 Open Repository Standards

Related to the issue of schema mapping is the establishment of semantic services for repositories which will enable the mapping of entities among application domains. The OMG has established simplistic repository services which are primarily focused on the registration and use of service signatures for validating and delivering object messages. The X3H4 committee (Information Repository Dictionary Standard, IRDS) and the ECMA Portable Common Tool Exchange (PCTE) effort have gone further in establishing semantic services. These services need to mature in order to mechanistically operate across overlaps in application domains.

Appendix A.

Glossary

- **AIM** — See Application Interpreted Model.
- **ANSI** — American National Standards Institute
- **AP** — See Application Protocol
- **Application** — A group of one or more processes creating or using product data.
- **Application Reference Model** — A component of a STEP application protocol. The high level information model that specifies the information included in an application protocol in the particular context of that application protocol. It describes the information requirements and constraints of a specific application context.
- **Application Activity Model** — A STEP model that describes an application in terms of its processes and information flow.
- **Application Context** — The intended use of product data within a business application. An application context is the unifying concept underlying a STEP Application Protocol.
- **Application Interpreted Model** — A component of a STEP application protocol. The mapping from the Application Reference Model (ARM) to the Integrated Resource Models (Parts 41-99; 101-199).
- **Application Protocol** — The unit of standardization in STEP (Standard ISO 10303). An application protocol is a suite of models that standardizes the semantics of a particular aspect of product representation at one or more stages in its life cycle. It describes the use of integrated resources satisfying the scope and information requirements for a specific application context.
- **Application Resource** — A STEP integrated resource whose contents are related to a group of applications as opposed to a "Generic Resource".
- **Architecture** — The documentation of the parts of some whole and their inter-relationships
- **ARM** — See Application Reference Model
- **CALS** — **C**ontinuous **A**cquisition and **L**ifecycle **S**upport. CALS is a Department

Appendix A.

Glossary

	of Defense initiative which is divided into two phases; 1) deliver electronic forms of information that is now delivered by paper, and 2) establish integrated databases of product description and support which spans contractors, subcontractors, and customers over the entire life-cycle of the product.
• **CITIS**	**C**ontractor **I**ntegrated **T**echnical **I**nformation **S**ystem.
• **EXPRESS**	Object-oriented information modeling language that is part of the STEP standard (ISO 10303-11)
• **Generic Resource**	A STEP integrated resource whose contents are completely context-independent as opposed to an "Application Resource".
• **IDEF0**	A graphical process modeling representation which is part of the US Air Force defined suite of IDEF tools.
• **IDEF1X**	A graphical information modeling representation which is part of the US Air Force defined suite of IDEF tools.
• **IGES**	Initial Graphic Exchange Standard. A standard which has been popular in the exchange of product description data. Its evolution has resulted in the addition of semantics, allowing the communication of meaning and intent rather than simple graphics. It will eventually be superseded by STEP in its role in exchanging product description data.
• **Integrated Resource**	A part of STEP that defines a group of resource constructs used as the basis for product data.
• **ISO**	International Standards Organization.
• **ISO 10303**	The official designation of the STEP standard.
• **NIAM**	A graphical information modeling representation which is popular in Europe and enjoys some degree of use in the USA.
• **PDM**	**P**roduct **D**ata **M**anagement
• **Product Data**	A representation of facts, concepts, or instructions about a product or set of products in a formal manner suitable for communication, interpretation, or processing by human beings or automated means.

Appendix A.

Glossary

- **RRM** Rapid Response Manufacturing Consortium
- **STEP** Standard for the Exchange of Product Data, ISO 10303.

Appendix B.

References

[1] Object Management Architecture Guide, Object Management Group

[2] Object Services Architecture, Object Management Group

[3] Common Object Request Broker Architecture, Object Management Group

[4] Hodges, Robert E, Chairman, ANSI X3H4.1, IRDS Technical Architecture Subcommittee, private communication.

[5] CALSIWG, "Profile for Enterprise Integration", CALS Industry Working Group whitepaper presented by the Profile for Enterprise Integration Working Committee at CALS Expo 1992.

[6] SEMATECH, "Realizing the Object-Oriented Lifecycle", presentation by Claude Baudoin and Glenn Hollowell of SEMATECH at Texas Instruments, 3 Nov 1993.

[7] Cattell, R.G.G., et.al., Object Database Standard: ODMG - 93, Morgan Kaufmann, 1993.

[8] Petrie, Charles J., editor, Enterprise Integration Modeling, Proceedings of the First International Conference, MIT Press, 1992.

[9] Zachman, John A., "A Framework for Information Systems Architecture," *IBM Systems Journal*, Vol. 26, No. 3, pp 276-292, 1987.