REFERENCE ARCHITECTURE for the
Intelligent Integration of Information

Prepared by the
Program on Intelligent Integration of Information (I^3)
Advanced Research Projects Agency

VERSION 0
April 18, 1995

Editors of this draft:

Richard Hull
Roger King

List of other contributors:

Michael Generesereth
Art Goldschmidt
Larry Kerschberg
Michael Siegel
Narinder Singh
Craig Thompson

Art Director:

Yigal Arens

Acknowledgement:

Ramesh Patil, for US Army Echelons of care example

CONTENTS page

FIGURES IN THIS DRAFT

Fig 2-1 : slide 3 from "Updated-Neches-foils, 1/27/95"
(same as slide 3 from the 1/25/95 version of these foils, I think)

Fig 2.2.1-1 : Three Layers of I3 Reference Architecture
(p. 1 of Mediation_Slides.ppt4, but updated to reflect current
high-level services)

Fig 2.2.1-2 : Service Descriptions Diagram
(this is the Neches box template -- p. 25 of
"Updated-Neches-Foils" from 1/27/95

Fig 3.1-1 : <Figure 3.1-1 is slide 1 of facilitation.p3>

Fig 3.3.1-1 : <FIGURE BROKER is slide 10 of facilitation.p3>

Fig 3.3.1-2 : <FIGURE BROKER-WIRING is slide 11 of facilitation.p3>

Fig 3.3.1-3 : <FIGURE BROKERING-EXAMPLE is slide 12 of facilitation.p3>

Fig 3.3.2-1 : <FIGURE FACILITATOR is slide 13 of facilitation.p3>

Fig 3.3.2-2 : <FIGURE FACILITATOR-WIRING is slide 14 of facilitation.p3>

Fig 3.3.2-3 : FIGURE FACILITATOR-EXAMPLE is slide 15 of facilitation.p3>

Fig 4.2-1 : Figure illustrating Ontology and Vocabulary Transf. Service

Fig 4.2-2 : Figure illustrating Context Transformation Service

Fig 4.2-3 : This and next one refer to the Description Service --
Please ask Mike what he has in mind for these

Fig 4.2-4 : see previous figure

Fig 4.2-5 : figure illustrating Comparator Service

Fig 4.2-6 : figure illustrating Resolver Service

Fig E.2-1 : <FIGURE ARCH showing agents, mediators, repositories,
facilitators, connected by ACL linkages>

Fig E.2-2 : <FIGURE PHYSICAL showing physical view with multiple
facilitators>

Fig E.2-3 : <FIGURE VARIATION -- direct connections to mediators,
other languages -- created manually or through
matchmaking services of facilitators>

Fig E.5-1 : <FIGURE AGENTS>

1) EXECUTIVE SUMMARY

The current I^3 Reference Architecture is designed to present an initial categorization of the principle "services" that can be used to support the intelligent integration of highly heterogeneous information sources. The reference architecture has three primary layers: Coordination, Mediation, and Wrapping Services.

One central problem being addressed is that of allowing large software applications to access in an integrated fashion a wide variety of legacy information sources on the "Information Superhighway". A natural result of any such effort is to develop standards for constructing and integrating such information sources. However, standards are only now emerging, and it is very expensive to convert legacy systems to such standards. Therefore, a primary focus of I^3 is on developing very high-level semantically rich standards that will make it easier to "retrofit" legacy information sources.

A second central problem is that choosing from the sometimes intractable number of available information sources and information intensive applications when attempting to satisfy a given information need. Therefore, another primary focus of I^3 is the development of mechanisms that can select and coordinate applications and information sources (legacy or otherwise) to accomplish a wide variety of information requests. In some cases the selection may be made at "compile-time", in the sense that it is static and made in advance. Or, the selection may be made at "run-time", and in this sense is dynamic.

Of course, many other organizations are addressing the issue of software interoperation. As a result, many of the concepts and services presented in I^3 Reference Architecture are not new. There are two important novel aspects of the architecture. First, the three-layer organization of services helps to isolate the broad range of I^3 issues, from the low-level interface and implementation issues, to very high-level issues involving data and program semantics. Second, and in particular, I^3 is focusing heavily on developing an integrated view of information integration services that are capable of reasoning about semantics.

2) MOTIVATION AND GOALS

There are two driving assumptions behind the ARPA-sponsored Intelligent Integration of Information (I^3) program. The first is that the "Electronic Superhighway" is an incredibly vast, but - as a direct result - an almost unusable information resource.

add: history of the I^3 reference architecture group, specifically
     the two workshops

The second underlying assumption is that information sources and information intensive systems are often semantically inter-related, but not in obvious and premeditated ways. The process of integrating two information sources can therefore be very complex.

The I^3 program involves researchers from the following institutions: xx,xx,..,xx. The various I^3 researchers are working in collaboration to develop software technology that can be used by information producers and consumers. The goal of I^3 technology is to signficantly reduce the time and complexity involved in sharing and integrating heterogeneous information on large-scale, widely distributed networks. (See figure 2-1.)

<FIGURE 2-1 : slide 3 from "Updated-Neches-foils, 1/27/95"
(same as slide 3 from the 1/25/95 version of these foils, I think)

2.1) GOALS OF THE REFERENCE ARCHITECTURE

The I^3 Coodination, Mediation, and Wrapping services are meant for the user of the emerging "Information Superhighway"; these users must be able to dynamically locate, access, and integrate a vast world of highly diverse information sources. The I^3 services thus support the development of large-scale information-based applications, and are used at application creation time. The resulting, generated I^3 "application" is really a runtime system that allows information intensive applications to coordinate their activities with multiple, legacy information sources. (But of course, new sources can be created at any time, and added to the set of deployed, "wrapped", ready-to-integrate information sources.)

Perhaps the most critical differentiator for I^3 technology is the high level of knowledge embodied in I^3 services. These services support semantically-rich, reusable, and evolvable interconnections between information sources and high level applications.

Following the document ‘‘An Architectural Framework for the National Information Infrastructure’’, generated by the XIWT (Cross Industry Working Team, which is developing an architectural framework for the NII), we adopt the following definition:

> "A reference architecture model describes a system in terms of the interconnection of basic functional elements and the interfaces between them. It clarifies where protocols must be defined and identifies groupings of functionality. It does not imply a physical implementation."

Thus, the I^3 reference architecture should be seen as a "vision" of how vast amounts of heterogeneous information can be incrementally pulled into a gigantic, reusable library of information resources. Thus, this document gives a concrete and detailed proposal for developing, classifying, and interconnecting tools that support large scale information integration. The focus of the I^3 reference architecture is to describe not the run-time operation of I^3 applications, but the three levels of services. We also do not focus on the programmer's view of using these services to generate new information intensive applications, but on the abstract classification of these services, and on the framework in which they are mean to be used.

2.1.1) Classses of Applications for I^3 Technology

There is an immense spectrum of applications that are a natural fit with the I^3 reference architecture. They include:

- logistics planning and support,
- health care information systems (specifically, the integration of patient medical records, along with blood serum results, xrays and MRI's),
- large scale software lifecycle environments that operate on wide-area networks,
- manufacturing, including engineering design
- scientific data
- international banking, and
- consumer activities (such as library searches and investment services).

It should be kept in mind that in the abstract, the I^3 reference architecture is meant to be application independent. Certainly, the I^3 services are less focused on wide area cooperative software systems, and more on large, centralized systems that use a wide variety of distributed information sources. But the architecture is not biased toward or focused on any specific class of applications.

Of course, as individual tools that provide I^3 services are developed, they will tend to focus on specific applications; this is simply a result of the need to include signficant application semantics in certain tools - not on the feeling that the architecture is something other than general purpose.

It should also be noted that certain I^3 reference architecture services will have to be developed to handle complex forms of information. Basic, "packaged" services are more likely to be oriented toward file-based, relational, or object-oriented data, and to be easily applicable to banking and other data processing activities. Multi-media, graphical, animated, textual, and sound-based applications (such as medical and library activities) will require much more specialized services.

2.1.2) Problems to be Solved

At its core, the I^3 reference architecture is addressing six specific problems that all relate to the immense difficulty in integrating diverse, legacy information sources, and in making them usable by large information based applications:

1 - The dramatic heterogenity among information sources:

Information sources are typically diverse at a number of semantic levels. Information may be stored in differing formats, such as flat files, relational databases, object-oriented databases, knowledge bases, or one-of-a-kind storage facilities. Schemas, vocabularies and ontologies differ - even when information from multiple sources share significant semantic relationships. The nature of various kinds of information may also vary signficantly, including text, images, audio, digital media, and complex engineering. Finally, the manner in which information is accessed may vary; this includes differing program interfaces, query languages, event handling protocols, and transaction mechanisms.

2 - The hidden and complex semantics in legacy information sources:

Often, application programs that create and maintain data are the key to understanding its intended use. In particular, in medical, insurance, banking, telecommuications, and other large applications, accessing legacy databases demands that we understand and re-use portions of large programs that define the interfaces to that data. These systems are likely to have embedded in them very complex rules concerning the interpretation and maintenance of this data, and these "rules" are often coded in ad hoc and difficult to interpret ways. In other words, it can be highly complex to deduce the intended semantics of legacy systems, especially since these systems can be extremely large and almost impossible to interpret externally. Further, these systems are often of highly varying natures, including knowledge-based systems, data processing systems, digital libraries, and software repositories.

3 - The need to build applications out of widely
    varying granularities of information sources:

It will often not be possible to make use of an entire information source. Internal components (such as selected segments of databases and specific procedures within within large software systems) will have to be wrapped and integrated with I^3 services. Simularly, very large and dense information sources (like entire libraries) will have to be made accessible.

4 - The large scale nature of many modern applications:

Not only are individual information sources large, but a single, new I^3 application may require the integration of a large number of information sources. For example, in order to implement a national heath care database, literally tens of thousands of information sources would have to be accessible in a coordinated fashion.

5 - The need to develop re-usable I^3 "building blocks":

Of course, with the almost unimaginable diversity and volume of existing information sources on the so-called Information Superhighway, it is critical that I^3 not significantly add to this complexity. In fact, I^3 tools must lessen this complexity. Thus, I^3 applications themselves, as well as the services used to generate these applications can be view as information sources, and must be built for easy re-use. It is this ability to develop libraries of deployed and re-usable I^3 applications that will gradually work to reduce the difficulty of managing the Information Superhighway's distributed information base.

6 - The evolution of information sources and I^3 services:

To make matters more complex, information sources, the programs that interpret them, and as a result, I^3 services and applications will constantly be evolving. This will call for the development of I^3 services that are much less opaque and much more understandable than the information sources they are typically integrating.

2.1.3) Intended Audience of the Document and Architecture

This document has three intended audiences:

1. The reference architecture is designed to communicate to the research, industrial, and military communities the focus and contributions of the I^3 program. In this way, we hope that the I^3 "vision" can have a significant impact on the development of a wide class of information intensive systems.

2. Within I^3, the reference architecture is meant as a way for individual researchers to position their work. This will facilitate the cooperation of the various researchers, and hopefully lead to a much more integrated and higher impact research program.

3. This document is also intended to lay a foundation for future ARPA programs, so that ARPA can have a cohesive, long term focus for information integration research.

2.2) OVERVIEW OF THE ARCHITECTURE

Subsection 2.2.1 introduces the three layers of I^3 services (Coordination, Mediation and Wrapping). It also defines the usage of the terms ‘service’, ‘tool’, and ‘I^3 application’ in the present document.

Subsection 2.2.2 discusses how the I^3 Reference Architecture can be used. Most of the issues considered here are out of the scope of the current phase in the design of the I^3 Reference Architecture.

2.2.1) Three Layers of Services

The I^3 reference architecture has three layers. They are indicated in Figure 2.2.1-1.

<FIGURE 2.2.1-1 - Insert gunning slide with 3 colored layers>

1. Wrapping Services: This layer corresponds to the raw information source level. Wrapping services are used to create common interfaces, to extract critical semantic information in order to facilitate integration with other sources, and to alter the structure, content and behavior of information sources in ways that are demanded by new I^3 applications.

2. Mediation Services: This layer involves the integration of individual information sources, in a way that takes advantage of semantic relationships among the information sources. Mediation services might unify two or more heterogeneous sources into a common source. They might keep multiple, heterogeneous information sources in sync as one or more of them are updated. They might in general,

serve any function that involves the semantic inter-relationships among multiple data sources. Mediation services are likely to communicate with "wrapped" information sources. Mediation services might also make use of other mediation services that provide smaller grained integration capabilities.

3. Coordination Services: Typically, services in this layer serve as a bridge between large scale information applications and information sources. These individual information sources have usually been "wrapped", and the semantic relationships between these sources are usually being controlled by mediation services. Coordination services serve such functions as locating useful sources of data, directing queries and updates to appropriate sources, and delegating tasks to individual mediator services and wrapping services.

In this document we distinquish between the notions of 'service', 'tool', and 'I^3 application'. Here, a tool is a software component that embodies one or more services. And a service is a conceptual functionality that could be supplied by one or more tools; no specific implementation is implied. In its current form, the I^3 Reference Architecture is focused primarily on identifying relevant services and their interrelationships. In this document we sometimes loosely refer to, e.g., the "input" or "output" of a service, or the services that a given service might use; in truth we are referring to the "input" or "output" of tools that implement the service. Finally, an I^3 application is a deployed software system that uses tools implementing I^3 services to provide integrated, semantically rich access to one or more information sources.

In a nutshell, the fundmantal difference between Coordination Services and Mediation Services is that Coordination Services are used to identify the available I^3 tools and underlying information resources that are needed to cooperatively get a job done, while Mediation Services provide the functionalities needed so that a given family of I^3 services and information resources can effectively interoperate. Wrapping Services perform well-defined syntactic transformations and mappings; this contrasts with Mediation Services which focus more on semantic transformations.

Sections 3, 4 and 5 of this document focus on the Coordination, Mediation and Wrapping Services, respectively. These sections present a listing of the high-level services in each layer, presented in a hierarchy based on the type/subtype relationship. When referring to specific services in this document we generally indicate its position in the hierarchy by giving its label. For example, the Data Query Services has position M.2.b. (We use 'C' for Coordination Services, 'M' for Mediation Services, and 'W' for Wrapping Services.)

Three perspectives are used in this document to describe various aspects of I^3 services. The first perspective is given in the type/subtype hierarchies for each layer. The second perspective is given through "is-used-by" relationships. These describe what services a given service might use. As a simple example, the Facilitator Service (C.3.b) might use the Smart Yellow Pages Service (C.1.c), among others.

The third perspective for describing services is the Service Description Diagram. Figure 2.2.1-2 shows the general form of these diagrams. A service might use three kinds of inputs: (1) repository knowledge, such as an ontology for a given application domain; (2) information from other services; and (3) data, this is the lowest granularity input, and loosely speaking is different with each "invocation" of the service. This diagram can show two kinds of output, one for the information produced by the service, and another focused on control, which corresponds loosely to requests that the service may make of other services or underlying information sources. The diagram can also represent the kinds of information that a service might keep in a local store (if any), and information about what services might control this service.

FIGURE 2.2.1-2 : Service Description Diagram
(this is the Neches box template -- p. 25 of "Updated-Neches-Foils" from 1/27/95)

When developing tools that implement I^3 services there is a tension between implementing the service(s) in a very narrow fashion vs. in a highly generic fashion. In some cases it is useful to develop tools that support I^3 services in a very generic way. For example, it would be useful for tools that implement the I^3 Inference Service (M.7.c.1) to be very generic, so that the same tool can operate with information from essentially any application domain, and using a broad variety of inferencing techniques (e.g., forward chaining, backward chaining, resolution, default reasoning, etc.) In terms of the Service Description Diagram, the Data input for such a generic inference tool might include a set of logical sentences (e.g., a rule base) along with an inferencing technique. The Repository Knowledge input might include information about a broad variety of inferencing techniques, giving relevant characteristics of them.

In other cases full genericity may not be necessary. Consider for example query optimization. While most query optimizers available today are focused on a single data model and a single query language, one can imagine optimizers that can work with queries from multiple query languages and data models. While this genericity with regards to query langauges may be warranted, it does not seem to be warrante in connection with data models. At present there are only five major data base models (hierarchy, network, relational, object-oriented, object-relational). Because this number is growing rather slowly, it is probably more appropriate to implement query optimizers that focus on these five models, rather than developing a much more costly generic optimizer which could handle all "possible" data models,

2.2.2) HOW TO USE THIS ARCHITECTURE

Several issues are beyond the scope of the current version of the I^3 Reference Architecture, and hence beyond the scope of this document. However, several of these issues provide critical context, and so we briefly touch on them here.

1. How are I^3 applications created out of I^3 services?

An I^3 application could be constructed by a tool that provides a coordination service; this service would support a general purpose methodology for selecting and integrating I^3 services and information sources. This is perhaps the most controlled way of developing an I^3 application. It is also likely to provide the highest level of re-usability of I^3 applications.

Or, an I^3 application could be constructed by a special purpose or ad hoc program. Finally, an I^3 application could be constructed by an entirely manual process, with a user interactively guiding the selection and integration of I^3 services and information sources.

It should be kept in mind that these are a few points along a spectrum. There are many possibilities that are in essence a combination of these three approaches. For example, an existing tool that provides a coordination service could be adapted in some ad hoc way. Or, this tool may require substantial human input.

2. What programming or software paradigm is used to create an I^3 application?

Clearly, there are going to be classes of I^3 applications, systems that share signficiant commonalities in the sorts of information they must integrate and the manner in which this information must be integrated. In essense, I^3 application developers are going to want to be able to re-use experience gained in developing individual applications. Sometimes, one I^3 application will simply serve as a building block for another. In the more general case, an existing application will need to be altered internally.

This issue is orthangonal to the first question, above. Whether the I^3 application is created by a tool coordination that supports a coordination service, or by a specialized program, or in a highly interactive way, the knowledge used to control the creation of the application should be stored in some re-usable manner.

There are number of ways in which this capability could be supported. A possible approach would be to use "templates", which would probably look like annotated graphs, that encode the functionalities of various I^3 services, the properties of various informations sources, and the useful interconnections of these services and sources. These templates could be grouped into classes, according to different sorts of I^3 applications, such as medical or banking applications. An individual template could be instantiated and possibily adapted as it is interpreted. Another approach would be to do the same thing with rule bases instead of templates.

Whatever approach is used, it would be a good idea to use some sort of quasi-declarative programming paradigm in order to enchance understandability and re-usability. But, it is critical to keep in mind that the number of possible templates or rule bases would be astronomical, given the extremely wide class of applications that might want to integrate diverse information sources. Thus, re-usability will in most cases be limited, and new I^3 applications will require fresh architectural work. But our goal is to incrementally develop large and organized libaries of templates or rules bases.

3. What interface protocols should be used for tools that support I^3 services? What langauge(s) should be used for communication between services within a layer, or across different layers?

Although I^3 applications will support the integration and interoperation of widely heterogenous information sources, it is important that the various tools supporting I^3 services can interoperate very easily. Perhaps the simplest way to support this is to adopt one or a small set of interface protocols for I^3 tools, and to adopt one or a small set of languages for communication between I^3 tools.

At a minimum, it appears that the CORBA standard developed by the OMG (see Section 2.3.1 below) can serve as a basic standard for I^3 tool interfaces and communication. This makes sense, because it would provide interoperability with many other information management tool developers. So, many I^3 functionalities should indeed be provided as OSA "object services". And, due to the specialized nature of many information sources, other I^3 services will likely be provided in a more application dependent fashion as OSA "common facilities" or OSA "application objects". And, it would seem natural, and probably mandatory, that the CORBA standard be used for communication within and between layers. But CORBA compatiblility - since it is really an implementation issue - deals with a level of detail below the scope of this document.

The agent paradigm (see Section 2.3.3 below) is another possible candidate for providing standard interface protocols for I^3 tools.

The issue of communication languages is closely related to that of communication protocols -- indeed, many communication protocols prescribe semantically rich languages. Many diverse and semantically rich kinds of information need to be communicated between I^3 services. The choice of one or more specific languages is beyond the scope of this document. Two candidates are suggested by the the interface protocols mentioned above -- the OMG IDL (see Section 2.2.1 and the KIF language that forms part of the Agent Communication Langauge developed by the ARPA Knowledge Sharing Effort (see Section 2.2.3)

Two final issues that are not addressed int his document are:

5. How do customers (humans or programs) interface with I^3 applications ?

6. What sorts of development environments exist for construction I^3 applications? These would include knowledge aquisition tools, modeling tools, and human interfaces for the development cycle.

Lastly, we note that this document gives a broad overview of the I^3 reference architecture; detailed discussions of various aspects of the architecture can be found in the following related documents

- pointers to related documents (e.g., detailed descriptions of various services) --

** Note: EVERYONE CAN CONTRIBUTE TO THE ABOVE LIST OF POINTERS TO RELATED DOCUMENTS

2.3) RELATIONSHIP TO OTHER ARCHITECTURAL APPROACHES

The I^3 Reference Architecture addresses a variety of issues that have also been addressed by other industrial and research consortiums, and also in the academic literature. This section considers the relationship of I^3 to the OMG architecture (Subsection 2.3.1), the NIIIP architecture (Subsection 2.3.2), the paradigm of agents (Subsection 2.3.3), and to a variety of other important architectures and paradigms that have been proposed (Subsection 2.3.4).

2.3.1) The Relationship Between OMG/OSA and the I^3 Reference Architecture

The Object Management Group (OMG) is an international non-profit organization supported by information systems vendors, software developers and users. OMG provides a widely supported framework for open, distributed, interoperable, scalable, reusable, portable software components based on OMG-standard object-oriented interfaces. Objectives of the OMG architecture are to benefit:

- application developers by making it much easier for developers to build large-scale, industrial strength applications from a widely available "toolkit" of standard components, and

- end-users by providing a "common semantics" that goes far beyond the "common look and feel" of today's GUIs.

The key idea underlying OMG is to develop a standard framework for communication between objects, and to develop standardized interfaces for objects that support a wide variety of services. The interfaces to objects supporting semantically rich services may themselves be quite semantically rich.

Because OMG is gaining increasing acceptance, it seems appropriate for I^3 to adopt the OMG standard. If this happens, then tools developed to support I^3 services would be immediately usable by a variety of non-I^3 tools.

As will be seen in Subsection 2.3.1.2 below, there is a broad overlap between the services that I^3 and OMG plan to focus on in the medium-term future. Thus, I^3 needs to work with OMG to minimize duplication of research and development focus. Indeed, the overlap between the family of services that OMG and I^3 are focusing on suggests that the I^3 program needs to carefully define its research agenda in order to be strongly differentiated from the OMG effort.

2.3.1.1 Brief Overview of OMG

This section gives a very brief overview of the OMG; more detail is included in Appendix D.1.

```
-----------------------      ---------------------
| Application Objects |      | Common Facilities |
-----------------------      ---------------------
```

```
           |                             |
 / --------------------------------------------------\
<                 Object Request Broker               >
 \ --------------------------------------------------/
                          |
                  -------------------
                  | Object Services |
                  -------------------
```

Figure 2.3.1-1 OMG Object Management Architecture

The {\italicize OMG Object Management Architecture Guide}, completed in 1990, serves as an overall architecture and roadmap for OMG standards development. It describes OMG's standard reference architecture, shown in Figure 2.3.1-1. The architecture consists of components, interfaces, and protocols which decompose an object system into four areas, classified as follows:

The OMG Common Object Request Broker Architecture (CORBA) specification provides a "message passing bus" for transporting requests and responses in a distributed environment. A principal component is the Interface Description Language (IDL), which is relatively imperative, object-oriented, and relatively language neutral.

Basic Object Services: These are services that most applications need, enapsulated as objects. A few examples are: Naming, Persistence, Concurrency and Transactions.

Common Facilities: These are a second collection of object services that provide general purpose capabilities, useful in many applications. A few examples are: compound documents presentation and interchange, Time Operations and Internationalization, Data Interchange and Automation Management, and an Agent Facility.

Application Objects: These are specific to particular end-user applications and are not subject to standardization.

2.3.1.2 Relationship of OMG and I^3 Reference Architectures

The goals of the OMG and I^3 are complimentary and have some overlap. As discussed below, several services that have been proposed by the OMG correspond closely to I^3 services described in this document. For this reason, current research into I^3 services has the potential of influencing the selection of services to be included into the OMG framework, and influencing the specification of their associated object interfaces.

There are several contrasts between the two architectures.
These include:
(1)
Typical I^3 applications may access information sources that are not OMG compliant; in this sense I^3 supports heterogeneity at a more fundamental level than OMG. On the other hand, it would be very natural to implement many of the I^3 services as OMG objects that use the OMG communication protocols.
(2) OMG is concerned with essentially all services that might be packaged as objects, whereas I^3 is focused more exclusively on services related to the integration of information; some of these may interact with other services and information sources in semantically very rich ways.
(3) OMG uses IDL as the primary langauge for communication between objects; in I^3 langauges with a more declarative flavor may be more appropriate in some cases.
(4) Finally, a modest contrast between the OMG effort and the I^3 Reference Architecture is that with I^3, there is an emphasis on understanding the interrelationship between I^3 services, along both class/sub-class and is-used-by dimensions; OMG is more concerned with identifying basic services and their interfaces, with little restriction on how they might be combined in particular applications.

There are two fundamental areas in which OMG and I^3 might interact:

(a) Some I^3 services might be implemented as OMG components (either as basic services, common facilities or application objects, depending on their level of generality), and

(b) Many of the basic object services and common facilities proposed or under development by OMG correspond closely to I^3 services identified in the I^3 Reference Architecture. This suggests that the two communities should cooperate in the development of standards for these overlapping services.

The remainder of this subsection considers these two issues in more detail.

In connection with implementing I^3 services as OMG components, we note that as detailed in Subsection 2.2.3, the current document is focused primarily on identifying the important I^3 services and the interrelationships between them (e.g.,, along the class/sub-class and is-used-by dimensions). A subsequent phase in the I^3 effort is to choose one or more paradigms for supporting relatively uniform interoperation between tools that implement I^3 services, including such issues as langauges for communication, standards for interfaces, and conventions for assembling composite services.

Because of its emphasis on supporting distributed interoperation between diverse components, the OMG framework is a strong candidate as the paradigm for the packaging of I^3 services, or at least for packaging some of them. The OMG framework provides a standardized repository service for IDL meta data, including message signatures, and provides a Program Invocation Service for registering and invoking new services.

An issue for further research concerns whether OMG's IDL is compatible with the communication needs of some I^3 services. Many I^3 services focus on the semantics of information, for example, involving integrity constraints, transformation specifications, metadata, knowledge and metaknowledge. In this context declarative langauges, possibly based on a relational data representation, are very useful. The KIF langauge (see Subsection 2.3.3) is a possible standard that might be used by I^3 services for communicating about these things. In contrast, OMG's IDL is largely imperative, and object-oriented, and does not lend itself well to this area.

We turn now to the overlap between proposed I^3 services and proposed OMG basic object services and common facilities. We give a high-level description of this overlap here; a more detailed description is presented in Appendix D.2. Areas of considerable overlap include

-- Information Transformation, including services for data transformation, (heterogeneous) distributed query processing,

-- Change Management, including version and configuration management

-- Simple Repositories: The CORBA Naming Service and Interface Repository are closely related to the I^3 White and Yellow Page Services.

In some areas the services provided by OMG and I^3 have similar spirit, although the I^3 analogs use more techniques from artificial intelligent. These include

-- Routing: OGM's CORBA provides basic routing services, whereas the I^3 content-based routing service can also select target objects based on the content of a request

-- Rules service: OMG is developing a service based on rules; I^3 currently plans the development of a more general Inference Service

-- Smart Yellow Pages Service: The OMG Trader Service is similar in spirit to, but not as sophisticated as, the I^3 Smart Yellow Pages Service

No OMG services so far proposed provide a framework for facilitation or negotiation.

OMG is evolving fairly rapidly, although there is typically a one year lag from RFP to adopted specification and another one year lag to commercial implementation. There is a strong opportunity for the I^3 program to influence the extension of OMG in the direction of I^3 technologies.

2.3.2) The Relationship Between the NIIIP and I^3 Reference Architectures

The National Industrial Information Infrastructure Protocols (NIIIP) facilitate the creation of Virtual Enterprises using "commercial off the shelf" (COTS) products. Virtual Enterprises (VEs) allow people in different organizations to rapidly develop a working environment to manage diverse resources toward the attainment of common goals. The typical goal is "profit maximization", broadly conceived; and the typical working environment is the design and manufacture of product in an industrial domain. Since many different products might be in use by the various members of a virtual enterprise, there is a need for common protocols into which the various products might map.

The application domain for NIIIP is the spectrum of industrial manufacturing enterprises and their supplier networks; it spans from the smallest shops to the largest corporations, and includes design, manufacture, and customer maintenance and support. For a simple product several dozen independent contractors might collaborate over the design specifications and manufacturing schedules. For a large project, such as Lockheed/Martin's F-22, thousands, or tens of thousands of component suppliers must collaborate on the assembly of perhaps a million part numbers. In both cases price, quality, and time of delivery are issues to be resolved, as are representation of data, tools (including choice of computer programs and algorithms), schedules and workflows to be utilized. Ultimately savvy decisions will require cooperative resolution among the members of the VE.

There is considerable overlap between NIIIP and I^3 goals in that both are developing reference architectures that focus on supporting the interoperation of multiple heterogenous information sources. As a result, several of the services involved in NIIIP are also present in the I^3 Reference Architecture. Furthermore, it is typical that at the formation stage, a virtual enterprise must rely heavily on mediation scenarios and searches to establish a common definition of the product and to establish common terminologies for the group's interoperation. There are also two primary differences between the two efforts:

-- In its current form the I^3 Reference Architecture is focused largely on the integration of heterogeneous information with an emphasis on providing read-only access, whereas NIIIP also focuses on the collaborative use of heterogenous information, as might arise in engineering design scenerios.

-- I^3 attempts to support the integration of information from essentially all information sources, whereas NIIIP favors standardization of interfaces to facilitate interoperation. In particular, the I^3 Wrapping Services can be used to modify legacy systems so that they appear to behave in more uniform ways. In contrast, NIIIP is also devoting substantial resources in working with particular standards bodies (including ISO, ANSI, OMG, I^3, and WFMC), and in developing a reference implementation of the architecture based on industrial scenarios.

-- I^3 attempts to support integration of information across essentially all domains, whereas the driving focus of NIIIP is on the construction of Virtual Environments in the manufacturing domain. (However, many NIIIP components may be useful in other domains.)

Because NIIIP is cooperating with and partially supported though the I^3 program, NIIIP plans to use tools developed by the I^3 community for those services needed by NIIIP that fall within the purvue of I^3.

This subsection first presents a brief overview of NIIIP, and then gives a comparison of the objectives of NIIIP and the I^3 Reference Architecture. More details concerning NIIIP and its relationship to the I^3 Reference Architecture are presented in Appendix E.

2.3.2.1) Brief Overview of NIIIP

The NIIIP protocols provide COTS products with the definition of component interfaces, described in the Object Management Group's CORBA Interface Definition Language (CORBA IDL). Once a COTS product supports some portion of this standard interface to the NIIIP protocols, it can then operate in concert with other such products to provide the functions prescribed for the operation of the Virtual Enterprise.

NIIIP protocols are grouped across 13 different component objects (presented in Section 2.3.2.3 below), to facilitate this packaging into OMG components. The component definitions are being submitted to the OMG as potential specifications within its Common Facilities architecture.

As another approach toward achieving a shared ontology amongst the members of the virtual enterprise, NIIIP is promulgating the use of

ISO standard STEP Product Data Modeling and Services. The adoption of such models and services simplify task of runtime mediation by having gained "up front" industry-level agreement on common terminology and compatible representations of product components. Standard models in automotive, electrical, drafting, 3-D configuration management, etc. are provided.

To account for business processes, NIIIP employs agent technology to model the organizational structure of the VE, and of all of the contributed resources within it. NIIIP agents representing organizational entities are concerned with time, quality, and price/cost. NIIIP agents representing resources are concerned with the best interests of the "resource". Since the interests of organizational components and resources will rarely converge, NIIIP prescribes a third party Negotiator to foster agreement. NIIIP will utilize standard I^3 technology and terminology to account for agent behaviors.

To support repeatable processes, NIIIP employes state-of-the-art workflow technology, and is working with the Workflow Management Coalition (WFMC) to develop and support standards development in this area. From the CAD Framework Initiative (CFI) NIIIP has adopted a robust Task and Session model, that coordinates the flows or work between disparate frameworks and products.

To facilitate wide-spread connectivity, the NIIIP builds upon many features of the Internet, enhancing them as needed, for example with new security features and object-orientation, to work within the "NIIIP Infrastructure".

2.3.2.2) Comparison of NIIIP and I^3 Reference Architectures

We compare the two architectures along four aspects: function, system administration, implementation paradigm, and the use of I^3 services within the NIIIP.

From a functional point of view, both architectures look to provide high-level mechanisms for simpliying customer access to information- and knowledge-based environements. NIIIP focuses more on supporting collaborative work in the industrial domain than in achieving a universally applicable paradigm such as I^3. I^3 architecture, conversely, supports more general requirements for heterogeneity and is focused on broader coverage over medical financial, marketing, academic, etc., as well as industrial, domains.

From a system administration point of view, NIIIP focuses on providing a family of standards and a core of code and information bases in the form of toolkits, classlibs, and "infobases" that facilitate the rapid bring up of virtual enterprise organizations. In this sense, the architecture, is geared toward deployment and distribution of technology. By contrast, in its current form the I^3 Reference Architecture specifies the general elements required in defining and administering integrated access to a multitude of heterogeneous information sources, but is less concerned with specific packaging for a given domain.

From the perspective of implementation paradigm, a key issue in the NIIIP project is settling on a broadly usable "object model". As of Cycle 0 in the NIIIP Spiral Development process, NIIIP objects are currently described in terms of amalgam of object modeling paradigms taken in the main from standard sources:

1. Interface (CORBA IDL)
2. Info Model (data content) (ISO STEP EXPRESS)
3. Rules (Event, Condition, Action)
4. Constraints (e.g. EXPRESS)
5. Associations (e.g.OMG Relationship Service, OSAM*, OpenOODBMS etc.)

This amalgam yields a high function object model. Its syntax is termed the NIIIP Common Language, and it is considered to be experimental. It supports the requirements of object modelers using ORBs, EXPRESS modelers, as well as Workflow Managers, Agents, Mediators, and Negotiators, and can be decomposed into its constituent standard sources (e.g. OMG CORBA IDL or ISO EXPRESS). At present the I^3 Reference Architecture does not prescribe a specific paradigm for implementing I^3 tools.

We now present a brief listing of the high-level I^3 services that NIIIP might use. A more detailed listing, that considers each of the 13 high-level NIIIP components and the I^3 services they use, is presented in Section E.2 of Appendix E.

The NIIIP protocols include the following high-level I^3 services:

Coordination Services:
-- Directory Services (C.1)
-- Management Services (C.2)
-- Some aspects of Content-Based Routing Services (C.3)

Mediation Services:
-- Several aspects of Information Integration Service (M.1)
-- Several aspects of Active Services (M.4)
this includes Workflow Services (M.4.c.4) (although the current intentions of NIIIP in this area appear to be more developed than those of I^3)

Wrapping Services
-- Data Restructuring Wrapping Service (W.2)
-- Application Program Modification Service (W.3.a)

The NIIIP protocols also address a variety of issues that are outside of the scope of the I^3 architecture, including:

- -- user interface issues
- -- network communication protocols and interfaces
- -- agents in a broad sense, for negotiation between services, and resolving issues such as ownership
- -- negotiation

2.3.3) AGENTS

The paradigm of agents provides a possible framework for implementing much or all of the I^3 Reference Architecture. Similar to the OMG protocols, the agent paradigm provides a set of guidelines and proposed standards for families of interoperating software modules and tools. Unlike the OMG protocols, the agent paradigm embraces languages that are more declarative that the OMG IDL (which is imperative and object-oriented), and may be better suited for transfering information about knowledge (e.g., rule bases) and metaknowledge (e.g., inference engines). This brief subsection gives an overview of the agent paradigm, and its relevance to the I^3 Reference Architecture. More details about the agent paradigm are presented in Appendix F.

In its most abstact form, the fundamental principle of the agent paradigm is that the "agents" must communicate with each other using a universally agreed upon language, called an "Agent Communication Language". The Knowledge Sharing Effort sponsored by ARPA has designed a specific Agent Communication Language, called here ACL, that has three components:

-- KIF: Knowledge Interchange Format

This is a variant of the first-order calculus with a LISP-like syntax. It includes features for naming, numbers, simple data strucctures, metaknowledge, limited forms of nonmonotonic reasoning, and some algebraic operations.

-- KQML: Knowledge and Query Manipulation Language

This provides a "conversation model" for the transfer of messages between agents. The message header may contain information about the intent of the message (e.g., a query, a reply, a command, a statement of fact), the language of the body of the message (e.g., SQL or KIF), the ontology used, etc.

-- Domain Ontologies:

It is anticipated that one or more ontologies would be developed for a given application domains, to form a common basis for communication about that domain.

Research into the relative advantages and disadvantages of using the OMG protocols vs. agent protocols as possible standards for the implementation of I^3 tools has not yet been performed. Speaking broadly, it appears that:

-- The agent paradigm supports more declarative forms of communication, and might thus be more appropriate for communication between most I^3 tools, since they will deal primarily with information and knowledge.

-- The OMG IDL is object-oriented, and may more easily represent a variety of important database concepts, including object identity, the closed world assumption (CWA), and complex types.

2.3.4) OTHER RELATED APPROACHES

** this is yet to be written, but could have very brief discussions of each related approach. Material for MRG's Facilitators and Gio's Mediators might be lifted from Section 6 of the March 29 version of Gio's glossary.

- JTF Architecture (crafted by Frederick Hayes-Roth and Lee Erman)
- federated databases
- schema integration
- repository technology
- process programming (software engineering point of view)
- CASE
- workflow management
- www
- MRG's Facilitators
- Gio's Mediators

2.4) A RUNNING EXAMPLE FOR THIS DOCUMENT: ARMY TRAUMA CARE

Throughout this document we will use one consistent application domain, to illustrate various I^3 services and how they can be used in concert.

2.4.1) Overview of Example Applicatoin Domain

This domain is focused on the information requirements for the US Army's "Echelons of Care" trauma care doctrine. Briefly, there are five Echelons of care:

- Echelon 1   Battalion level and below treatment
  Life-saving first aid to permit evacuation to Echelon 2

- Echelon 2   Medical Company level treatment
  Emergency medical treatment, with whole bolld capacity, but generally no surgery

- Echelon 3   Division level treatment
  Resucitation, Emergency wound surgery, post-operative treatment

- Echelon 4   Definitive treatment outside theatre of war
  Completely outside of war zone; full hospital service

- Echelon 5   Long term and rehab care outside theatre of war
  Hospitals in the US

An Echelon 2 medical company is typically located in friendly territory, and staffed by six to twelve physicans, along with support staff. A primary function is to perform emergency surgery, in order to stabilize patients and protect limbs. If recovery is expected to take more than 72 hours, then the patient will be evacuated to an Echelon 4 or 5 facility.

The brigade and divison level Health Services Support function are are responsible for medical planning, readiness, care and reporting.

More detailed information on the five Echelons can be found in Appendix C.

2.4.2) Information Sources

A very wide variety of information sources are used in this application domain. The information sources are very heterogeneous, invovling different ontologies, different schemas, different database management systems. Further, some information sources are not digital, and the semantics of others is buried in application software. The following lists some of the sources used in examples of this document:

- Casualty and sick call reports, prepared nightly by medical officers in the first and second Echelons.
- Medical assets information, determiend from troop movement reports, requests for supplies and from status reports concerning personel and supplies.
- Blood system databases.
- TRACCES information system; this is an army-wide system for managing logical information about transportation. (This would include the location of planes and trucks, and the availability and location of fuel and airports.)
- Field medical records
- Medical records from home bases; although these generally use the Army-standard CHCS (Composite Health Care System) DBMS, the schemas typically vary considerably.

3) COORDINATION SERVICES

** Note: we need to include citations to the literature
where these concepts/services first appeared

3.1) Overview and Crisp Definition

The Coordination layer in the I^3 reference architecture is concerned with providing a collection of communication and coordination services. These services provide a bridge between applications and information resources, as shown in Figure 3.1-1.

<Figure 3.1-1 is slide 1 of facilitation.p3>

The applications typically make requests of the services provided by the coordination layer. The services in the Coordination layer make use of the information resources to service the requests from the customer applications. Note that the customer/information resource distinction is not fixed. That is, it is possible for a customer to be an information source (e.g., a human at a decision support system can also answer questions), and it possible for an information resource to make requests (e.g., for a simulation system to request the services of a numerical integration library in simulating the analog behavior of transistors).

3.2) High-Level Coordination Services

In this subsection we focus on the Coordination Services and describe their utility, the different subtypes of the services, and the relationship between the services (e.g., which services make use of other simpler services).

The high-level Coordination Services are now listed.

C.1: Directory Services
  C.1.a: White Pages
  C.1.b: Yellow Pages
  C.1.c: Smart Yellow Pages

C.2: Management Services
  C.2.a: Program Invocation
  C.2.b: Scheduling and Sequencing
  C.2.c: Matchmaking

C.3: Content-Based Routing

  C.3.a: Broker
  C.3.b: Facilitator

Each of these services is now described in greater detail.

C.1: Directory Services

The directory services provide a collection of services for getting information about the applications and information sources that are available, and the specific services that they can provide directly. The specific directory services are White Pages, Yellow Pages and Smart Yellow Pages.

These services are useful for identifying the available I^3 tools in an I^3 application, and to identify the tools that can handle a particular request at run-time. In a dynamic setting, the set of I^3 tools can come and go, and the directory services are essential to check if an I^3 tool is still available at run-time, and to search for tools based on the services they provide, instead of searching for I^3 tools by name. Searching for I^3 tools by capability is preferable to compiling in the choice of I^3 tool to use. An I^3 tool may no longer be available at run-time, yet the services it provides may be provided by another. In addition, a new I^3 tool may provide the same services better than a previously existing known I^3 tool can. In both these situations, the directory services make the system more flexible and robust.

C.1.a: White Pages

The white pages service acquires and maintains dynamic information about I^3 tools and their status. A request can be made to check if a specific I^3 tool is registered, to check the active state of a specific I^3 tool (active or inactive), or to get a list of all the I^3 tools. In addition, the white pages services can be instructed to register, unregister, activate, or deactivate an I^3 tool.

The white pages services is primitive, i.e., it records the information about the I^3 tools and their status directly.

The inputs to the white pages service module are:
 o service ontology (the communication ontology)
 o request (is an I^3 tool registered, active, what are the names of all I^3 tools)
 o instruction (register, unregister, activate, deactivate) from service
   providers

The output of the white pages is the list of I^3 tools satisfying the request.

C.1.b: Yellow Pages

The yellow pages service acquires and maintains dynamic information about I^3 tool names and the types of services that they provide. The yellow pages can be queried to find out the names of all I^3 tools that can provide a service of a particular type (e.g., print a file, answer a query). The yellow pages service answers its queries against the database of capabilities information that it maintains. In answering the queries, the yellow pages services looks for a match between a query and the capabilities of the I^3 tool services that it has recorded. The name of the I^3 tools with capabilities that match the query are reported, if any such I^3 tools exist. Note that the matching of requests against capabilities is a simple pattern match, i.e., it does not involve any translation or inference.

The services used by the yellow pages are:
- o ''knowledge repository''

The knowledge repository is used to record the services provided by I^3 tools, i.e., their capabilities.

The inputs to the yellow pages service module are:
- o service ontology (the communication ontology)
- o request (service pattern- what I^3 tools can provide such a service)
- o instruction (dynamic information about I^3 tools and the services they provide)

The output of the yellow pages is a list of I^3 tools that can service the request.

White pages and simple yellow pages directory services are provided by many existing distributed computing environments, e.g., by the various CORBA complient systems. The possible difference is that in I^3, more expressive langauges based on first-order logic might be used for specifying the capabilities of I^3 tools and information sources. In most traditional systems the capabilities are specified in an Interface Description Language (IDL), which defines the patterns of requests that are handled (with type restrictions on fields). A simple pattern matching algorithm, with type checking, is used to search for an object that can handle a request. In contrast, I^3 White and Yellow Pages can support the matching of requests to services in I^3 is based on declarative descriptions and inference, which is much more powerful, e.g., it can be used to search for I^3 tools that can handle a request and in addition have certain attributes.

C.1.c: Smart Yellow Pages

The smart yellow pages service is similar to yellow pages. It makes use of the yellow pages service, and it acquires and maintains dynamic information about a domain. The domain information is used in combination with the yellow pages service to provide more powerful services. The smart yellow pages can be queried to find out the names of all I^3 tools that can provide a service of a particular type. The more powerful service is provided by using an inference procedure to match the query against the capabilities of I^3 tools by making use of the domain theory.

The services used by the smart yellow pages are:
- yellow pages
- knowledge repository
- inference system

The knowledge repository is used to record the domain information and the inference system is used to service the requests.

The domain theory records facts and rules. This information can be used, for example, to define properties of I^3 tools and the relationship between different concepts. For example, it can be used to define the relationship between rectangular and polar coordinates.

Inference is used to overcome syntactical mismatches between a request and the recorded capability of an I^3 tool or information source. For example, suppose an I^3 tool provides the service of vector addition in polar coordinates, the domain knowledge defines the mapping between rectangular and polar coordinates, and a request is made to find out the names of all I^3 tools that can add two vectors in rectangular coordinates. The simple yellow pages will fail in this case because of the syntactic mismatch between the request and recorded capabilities. The domain theory defines the semantic equivalence independent of the syntactic mismatch. The smart yellow pages can take advantage of this to translate the requested service, and identify the I^3 tools that can handle the translated request.

The inputs to the smart yellow pages service module are:
- service ontology (the communication ontology)
- request (service pattern- what I^3 tools can provide such a service)
- instruction (dynamic information about domain knowledge)

The output of the smart yellow pages is a list of I^3 tools that can service the request.

C.2: Management Services

The management services provide a collection of services for invoking and scheduling I^3 tools. In addition, they provide services for matchmaking, i.e., services to design, set up, and manage configurations of services to handle requests of a given type. The specific management services are: Program Invocation, Scheduling and Sequencing, and Matchmaking.

These services are useful for managing and controlling the activities of I^3 tools. The invocation services are useful for starting up I^3 tools that provide a given service. If the service requires a separate program thread per request, or if the service is used infrequently, it must be automatically invoked when it is needed. The scheduling and sequencing services are used to control the order in which complex services are handled. For example, for a conjunctive service, the scheduling and sequencing service selects the order in which the component service are to be called. The first two are support services. They are automatically invoked by the system in the process of handling a request from an application. The matchmaking service is useful for automatically designing and setting up communication links between applications and information providers to handle a class of requests. Once matchmaking has been set up, the system is completely out of the loop, and the applications and information resources can communicate directly, possibly in a specialized language. This has the advantage of greater efficiency by using specialized languages and by removing the system from all interaction. The matchmaking service can be used, for example, to set up the connection between an application displaying NYSE data with an information source with video data for it.

Automatic invocation of programs is supported in some distributed systems environments. The scheduling and sequencing services are provided by query optimization services in data bases. The work on automatic matchmaking is novel, and builds upon the work done in automatic programming and planning. The matchmaking service is much simpler than automatic programming, since it only involves generating the glue to connect existing programs. It can be as simple as directly connecting two I^3 tools, yet it can require more complex configuration designs involving multiple I^3 tools.

C.2.a: Program Invocation

The Program Invocation Service is used to automatically invoke the execution of an I^3 tool in order to service a request. This is a service that is not directly visible to a requester of a service, but is invoked automatically in order to handle a request. This service is similar to the automatic Program Invocation Service provided by a HTTPD daemon in the World-Wide Web (WWW).

The input to the program invocation module is:
 o the name of the I^3 tool to be activated.

The output of the program invocation module is:
 o the control commands to activate I^3 tools

The Program Invocation Service is primitive and does not make use of any other services.

C.2.b: Scheduling and Sequencing

The Scheduling and Sequencing Service provides meta-level control capabilities. This service can be used to select the order of activation of component services that are used in order to handle a complex request. For example, for a conjunctive request, the order in which the conjuncts are handled can have a tremendous impact on efficiency (e.g., finding all residents of California that were presidents of the U.S.). This is a service that is not directly visible to a requester of a service, but is invoked automatically in the process of handling a request.

The input to the Scheduling and Sequencing Service is:
- o the service requested

The output of the Scheduling and Sequencing Service is:
- o the schedule of activations for the more primitive services

The services used by the scheduling and sequencing module are:
- o Knowledge Repository Service

The Knowledge Repository Service records the domain information for the primitive services (e.g., there are far more residents of California than there are U.S. presidents).

C.2.c: Matchmaking

The Matchmaking Service designs, sets up, and manages the configuration of services to provide a requested service. The Matchmaking Service is given a requested service, the directory of services provided by existing I^3 tools, and a repository of domain knowledge that provides the definitions and relationships between concepts. The Matchmaking Service designs a configuration to achieve the requested service. The configuration defines the components used, the communication links between the components, any translation of data on the communication links, and a control program for sequencing requests. For example, the Matchmaking Service can be used to set up a communication channel between the requester of a service and a provider. In this simple setting there is no translation required. In a more complicated setting a NYSE information provider and a Nasdaq information provider can be connected together with a simple control program to provide stock information from either market. The Matchmaking Services are in effect run-time mediator generators.

The services used by the Matchmaking Service are:
- o configuration manager
- o configuration designer

- o directory of existing services
- o domain knowledge repository

The Configuration Manager Service orchestrates gets requests for matchmaking, calls the Configuration Designer Service to create a design, and then executes the design. The Configuration Designer Service uses the Directory of Existing Services Service to find out the primitive services that are available. It also uses the Domain Knowledge Repository Service to combine the primitive service to achieve the desired service.

The inputs to the Matchmaking Service are:
- o service ontology
- o service request
- o data about service providers

The output of the Matchmaking Service is:
- o linkage control to configure and control services

C.3: Content-Based Routing Services

The content-based routing services provide one-stop shopping capabilities for applications. Users send information and/or requests to the content-based routing services, which in turn delegate to appropriate services. The specific content-based routing services are: Broker and Facilitator.

The content-based routing services make use of a meta-data repository of the capabilities and interests of other I^3 tools in order to handle user requests. The content-based routing services examine the request, select the I^3 tool(s) that can handle request (based on their meta-data), send the requests to the appropriate I^3 tools, get the answers, and return the result to the user.

The content-based routing services are important because they remove the burden of interoperation in a dynamic environment from a user to the system. The services provided are applicable to all users, and this simplifies the process of creating applications. Users need not be concerned with the available I^3 tools at run-time, the capabilities of the I^3 tools, selecting the appropriate I^3 tools by decomposing a complex request into its component sub-requests, translating sub-requests into a form understood by the I^3 tool providing the sub-service, gathering the results for the sub-requests, and combining these to obtain the desired service.

The impact of content-based routing is to give the illusion that it directly handles all requests. From a users point of view there is considerable reduction in complexity, since there is no need to know about any other I^3 tools, their capabilities and interests, or any domain information relating to them. A user is unaware of how the content-based routing services handle a request, and it is unnecessary for them to know so. The disadvantage of the content-based routing services is that all decisions are made at run-time. In certain situations it is possible to increase efficiency in a static world by making the decisions once at compile-time, which is in effect what is provided by mediation services.

Simple yellow-pages services are provided in distributed computing environments, e.g., CORBA. These are used to select an object based on simple pattern matching between a request and interface specifications in IDL. The user must actually make the request and obtain the answer directly from the object that can handle the request. The SHADE system extends this by providing content-based routing without translation or problem decomposition. This removes the burden from the user to actually make the request to the I^3 tool that provides the service. The Agent-Based Software Interoperation (ABSI) system extends this significantly by including problem decomposition and translation in the content-based routing services that it provides.

C.3.a: Broker

The Broker Service provides simple content-based routing services. A customer sends a request to a (tool implementing a) Broker Service, and the Broker Service uses a meta-data repository to search for an I^3 tool that can handle the request directly. The request is sent to an I^3 tool that can handle the request, and the answer returned by the selected I^3 tool is returned. The Brokering Service does not perform any translation or problem decomposition. Such requests are handled by other I^3 Mediation and/or Wrapping services. In addition, the Brokering Service does not support conditional capability descriptions for I^3 tools. For example, descriptions that depend on attributes of the request, as in an I^3 tool that specifies it can provide information about the stock price for any company, as long as the company's stock is traded on the New York Stock Exchange. Nor do the brokering services support conditional requests. For example, requests that specify certain attributes for the I^3 tool servicing the request, as in requiring that the cost of the service be less than some threshold.

The services used by the Broker Service are:
- o yellow pages
- o communication manager

The Communication Manager Service coordinates the requests and replies from the I^3 tools that service the request.

The inputs to the Broker Service are:
- o service ontology
- o request (the service to be performed)
- o meta-data about services
- o response from I^3 tool providing service

The outputs of the Broker Service are:
- o control input to I^3 tool providing servicing the request
- o response back to the user

C.3.b: Facilitator

The Facilitator Service is similar to the Broker Service, but it provides more powerful services. The Facilitator Service uses translation capabilities to enable routing in a wider class of cases. In addition, the Facilitator Service can also decompose a complex query into a collection of sub-queries, select the sequence of execution for the sub-queries, select the I^3 tools that can handle the sub-queries, transmit the sub-requests to I^3 tools, get the sub-results, and finally combine these to get the answer. This will use the Distributed Query Generation Service (M.1.c.3), but is richer because of the possibly dynamic nature of the I^3 tools and information sources available at a given point in time. The Facilitator Service uses inference to provide these services, and the Inference Service (M.7.c.1) also permits conditional capability descriptions for I^3 tools and conditional requests.

The services used by the Facilitator Service are:
- o smart yellow pages
- o scheduling and sequencing
- o communications manager

The inputs to the Facilitator Service are:
- o service ontology
- o request (the service to be provided)
- o meta-data about services
- o responses to sub-requests

The outputs of the Facilitator Service are:
- o control inputs to the I^3 tools servicing the sub-requests
- o response back to the user

3.3) Examples

** These examples are not from the Army Trauma Care domain.
   They will be replaced.

This subsection presents two examples illustrating the Broker and

Facilitator Services.

3.3.1) Broker

The brokering scenario illustrates content-based routing for a simple request. The user wishes to find out the price of IBM stock on the 23rd of September 1994. The user sends the request to the broker, and expects the broker to handle the steps required to service the request on its own and pass back the result.

A high level picture of the broker is given in Figure BROKER (3.3.1-1), which shows the top-level inputs and outputs.

<FIGURE BROKER is slide 10 of facilitation.p3>
[Fig 3.3.1-1]

The wiring diagram of the broker, showing its constituent services, is given in FIgure BROKER-WIRING (3.3.1-2).

<FIGURE BROKER-WIRING is slide 11 of facilitation.p3>
[Fig 3.3.1-2]

The instantiation of the general wiring diagram for this example is shown in Figure BROKERING-EXAMPLE (3.3.1-3).

<FIGURE BROKERING-EXAMPLE is slide 12 of facilitation.p3>
[Fig 3.3.1-3]

The user request is first handled by the communications manager, which uses the yellow pages service to find out that the New York Stock Exchange tool can handle this request. The communications manager passes on the request to the New York Stock Exchange tool, which replies that the price of IBM stock on the 23rd of September, 1994 is $56. Finally, the communications manager returns this answer to the user.

3.3.2) Facilitator

The facilitator scenario illustrates content-based routing for a more complicated request, where the information cannot be provided by a single I^3 tool. In this example the user wishes to find out the average number of employees of the fifty biggest companies (in terms of value) on the Silicon Valley Technology Index (SVTI). The user sends the request to the facilitator, and expects the facilitator to handle the steps required to the request on its own and pass back the result.

A high level picture of the facilitator is given in Figure

FACILITATOR (3.3.2-1), which shows the top-level inputs and outputs.

<FIGURE FACILITATOR is slide 13 of facilitation.p3>
[Figure 3.3.2-1]

The wiring diagram of the facilitator, showing its constituent services, is given in Figure FACILITATOR-WIRING (3.3.2-2).

<FIGURE FACILITATOR-WIRING is slide 14 of facilitation.p3>
[Figure 3.3.2-2]

The instantiation of the general wiring diagram for this example is shown in Figure FACILITATOR-EXAMPLE (3.3.2-3).

<FIGURE FACILITATOR-EXAMPLE is slide 15 of facilitation.p3>
[Figure 3.3.2-3]

In this example, the companies in the SVTI are traded on different stock exchanges- NYSE, Nasdaq, and the Chicago SE. These stock exchanges are permanent, and there is a SVTI mediator that handles all information about stock values, number of shares outstanding, etc. for all companies in the SVTI. There is also an IRS tool that keeps track of the number of employees in a company, their corporate tax information, etc.

The user sends the request to the facilitator, and request is first handled by the communications manager. The communications manager, using domain knowledge, decides that it must first find out the valuation of the companies in the SVTI to find the top fifty, and it must then find out the average number of employees for these. It uses the smart yellow pages to find out that the SVTI mediator knows the members of the SVTI, what their current stock price is, their number of shares outstanding, etc. The communications manager sends the request to the SVTI to find out the stock price and number of shares outstanding for all the companies in the SVTI. It uses this information to find out the top 50 valued companies (using domain knowledge to compute the value of a company from its stock price and number of shares outstanding).

The communications manager must next find out the average number of employees for these top 50 companies. It uses the smart yellow pages to find out that the IRS tool can answer questions about the number of employees for each of the top 50 companies in the SVTI. The manager sends a request to the IRS tool for each company to find out its number of employees. It uses this information to compute the average (using domain knowledge about the definition of an average), and finally passes the answer back to the user.

4) MEDIATION SERVICES

** Note: we need to include citations to the literature where these concepts/services first appeared

4.1) Overview and Crisp Definition

The role of mediation services is to provide a middle layer of services situated between the coordination services and wrapping services. Mediation services play a crucial role in the I^3 reference architecture because they provide those service which integrate data >from multiple sources and provide the infrastructure for users, programs, and agents to integrate diverse information, share information with varying meaning and contexts. Mediation services enable users to incorporate legacy databases into their information architecture and provide an integrated view of the enterprise information resources.

4.2) High-Level Mediation Services

The mediation services are divided into seven major categories: 1) Information Integration Services, 2) Query Services, 3) Multistate Services, 4) Active Services, 5) Explanation Services, 6) Resource Management Services, and 7) Elemental Services. Within each category, the services are further divided into three broad subclasses: Schema-related services which deal with meta-data issues such as data schemas, constraints, and knowledge, Data-related services which deal with the actual data and their formats, types and related issues, and Control-related services which address the control and operational aspects of providing a service.

The complete outline of the major Mediation services is now given. After that, we present each major service category, a short description of that category, and then the constituent services of that category. Also presented are some of the most important is-used-by relationships of a service to other services (which arise if one service can make requests against another service).

```
M.1: Information Integration Services
  M.1.a: Schema Integration Services
    M.1.a.1:   Ontology and Vocabulary Transformation Service
    M.1.a.2:   View Transformation Service
    M.1.a.3:   View Integration Service

  M.1.b: Data Integration Services
```

M.1.b.1: Context Transformation Service
M.1.b.2: Abstraction and Aggregation Service
M.1.b.3: Object Correspondence Service
M.1.b.4: Inconsistency Management

M.1.c: Access Policies Services
M.1.c.1: Distributed Update Generation Service
M.1.c.2: Materialized and Virtual View Management Service
M.1.c.3: Distributed Query Generation Service
M.1.c.4: Query Result Integration Service
M.1.c.5: Caching Service

M.2: Query Services
M.2.a: Schema Query Services
M.2.a.1: Browsing meta-data
M.2.a.2: Browsing meta-knowledge
M.2.a.3: Formulation Services
M.2.a.4: Reformulation Services
M.2.a.5: Semantic Heterogeneity Resolution
M.2.a.6: Relaxation Services
M.2.a.7: Association Services

M.2.b: Data Query Services
M.2.b.1: Browsing Data Services
M.2.b.2: Browsing Knowledge Services

M.2.c: Control Query Services
M.2.c.1: Query Planning Services
M.2.c.2: Query Optimization Services
M.2.c.3: Multiple Query Optimization

M.3. Multistate Services
M.3.a: Schema Multistate Services
M.3.a.1: Schema Version Service
M.3.a.2: Time-Stamp Service

M.3.b: Data Multistate Services
M.3.b.1: Data Version Service
M.3.b.2: Replication Service
M.3.b.3: Hypothetical Access Service

M.3.c: Control Multistate Services
M.3.c.1: Version Management Service
M.3.c.2: Replication Management Service

M.4: Active Services
M.4.a: Schema Active Services
M.4.a.1: Active View Materialization Services

```
  M.4.b: Data Active Services
    M.4.b.1:   Alerting Services
    M.4.b.2:   Monitoring Services
    M.4.b.3:   Subscription Services

  M.4.c: Control Active Services
    M.4.c.1:   Event-Condition-Action Rule Execution Services
    M.4.c.2:   Update Triggering and Propagation Service
    M.4.c.3:   Rule Conflict Resolution Strategies
    M.4.c.4:   Workflow Services

M.5: Explanation Services
  M.5.a: Schema Explanation Services
    M.5.a.1:   Schema Integration Explanation
    M.5.a.2:   Schema Transformation Explanation
    M.5.a.3:   View Explanation

  M.5.b: Data Explanation Services
    M.5.b.1:   Data Format Explanation
    M.5.b.2:   Data Lineage Explanation

  M.5.c: Control Explanation Services
    M.5.c.1:   Mediation Process Explanation
    M.5.c.2:   Coordinator Process Explanation
    M.5.c.3:   Wrapping Process Explanation
    M.5.c.4:   Inference Explanation

M.6: Resource Management Services
  M.6.a: Schema Management Services
    M.6.a.1:   Repository Support Services
    M.6.a.2:   Directory Support Services

  M.6.b: Data Management Services
    M.6.b.1:   Ontology Support Services

  M.6.c: Control Management Services

    M.6.c.1:   Interface Services
    M.6.c.1:  (1)    Facilitator Reporting Services
    M.6.c.1:  (2)    Wrapping Calling Services
    M.6.c.2:   Cost Estimation

M.7: Elemental Services
  M.7.a: Schema Elemental Services
    M.7.a.1:   Description Service
    M.7.a.2:   Comparator Service
    M.7.a.3:   Resolver Service
```

M.7.b: Data Elemental Services
  M.7.b.1: Description Service
  M.7.b.2: Comparator Service
  M.7.b.3: Resolver Service

M.7.c: Control Elemental Services
  M.7.c.1: Inference Service
  M.7.c.2: Ontology Service
  M.7.c.3: Database Service

xxx at some point Neches-boxes should be generated for some of these.

4.2.1 Brief Descriptions of the Major Mediation Services

M.1: Information Integration Services

These services support the transformation and integration of schemas, knowledge, data from multiple sources. (It should be noted that in some cases there is only one information source; this occurs, for example, when the integration base is simply a derived source that must be kept in sync with a different information source.) Data abstraction and summarization services allow for the aggregation of data into meaningful units for presentation to decision-makers. The context of information is important to establish guidelines for its use, and the role of the information within the context of one or more applications.

M.1.a: Schema Integration Services

This set of sub-services control the mapping of information source schemas into schemas for integration bases.

M.1.a.1: Ontology and Vocabulary Transformation Service

The ontology and vocabulary transformation service is used to take multiple shared ontologies and produce a single ontology that combines common aspects between the original ontologies. This service is very useful when information is to be moved between different operations where different ontologies may be used. For example, to move information from a hospital operation (e.g., medical ontology) to battlefield operation (e.g., battlefield ontology) the ontologies of each of these operations must be merged in such a way that common concepts are clearly defined. The hospital operation may refer to patients by civilian name and may give health statistics for that individual, while battlefield operations might refer to individuals by military identifications and store aggregate health statistics. Moving information from hospital records to battlefield operations will require an identification of overlapping concepts (i.e., using the OVTS) and ultimately a reconciliation of the differences in structure and meaning of this information . The intention of the OVTS is to provide common concepts identifiers and apply necessary transformations to support these common concepts. In the hospital/battlefield example, both ontologies would be sent to the OVTS (i.e., Ontology 1, Ontology 2) with their mappings to some Service Ontology (i.e., set of shared ontologies). The output would be a new ontology that described the shared concepts (Ontology 1) in terms of the battlefield terminology (Ontology 2). (See Figure 4.2-1.)

FIGURE 4.2-1 GOES HERE

Projects Providing Ontology Transformation Services include: MIT, GMU, SIMS, Co-Base, etc.

M.1.a.2: View Transformation Service

This service allows a local database schema to be mapped into the objects supported by the common data model.

M.1.a.3: View Integration Service

This service maps source schemas expressed in the common data model to be integrated into the "integrated" schema.

M.1.b: Data Integration Services

This set of sub-services control the logical integration of data from multiple sources into a common integration base (or one source into a restructured target).

M.1.b.1: Context Transformation Service

The transformation service can translate a term and its context from a source ontology to a destination ontology by applying the appropriate ontological transformation knowledge. Examples of sources and destinations include databases, knowledge bases (e.g., ontologies) applications, sensors, and humans.

The context transformation service block diagram is shown in Figure 4.2-2 below. The inputs to the context transformation service include an ontology, the source context definition and the data being exchanged, the receiver's context definition, a set of mappings from each of the two contexts to the ontology, and a control function. The result of context transformation is data converted to the desired context and knowledge (i.e., meta-data) describing the resulting context. Additional output might include knowledge about the difference between the local contexts (i.e., semantic conflicts) and a description of the process used to resolve the conflict. In cases where conflicts cannot be resolved the service returns an error code and description.

FIGURE 4.2-2 GOES HERE

Context transformation is a complex service that can be described in terms of more elementary services. These services include a Descriptor Service, a Comparator Service, and a Resolver Service.

A special kind of context transformation concerns temporal discrepancies among data in several information sources. These arise from differing temporal "granularities" used to measure time units. A temporal ontology is used to express temporal concepts and their relationships, for all temporal units, and appropriate temporal transformations are applied to data to provide an integrated temporal result to a query. This service would make use of the context transformation service described in 2.a above.

M.1.b.2: Abstraction and Aggregation Service

This service allows users and applications to specify abstractions, aggregations, and summarizations of data from multiple sources into an integrated "picture" at the common level.

M.1.b.3: Object Correspondence Service

This service allows objects from distinct information sources to be specified as being the same object instance in the common schema.

M.1.b.4: Inconsistency Management

This service supports the resolution of data inconsistencies resulting from a query to several databases. The conflicting answers are analyzed with the respect to their goodness and the reliability of the information sources to produce the "integrated" answer to the query.

M.1.c: Access Policies Services

This set of sub-services controls the policies used for physically managing the integration base and the multiple sources. These policies include the management of accesses (including both queries and updates) from sources to the integration bases and the management of accesses from the integration base to the sources.

M.1.c.1:   Distributed Update Generation Service

This determines necessary updates to multiple sources, given logical updates to integration bases.

M.1.c.2:   Materialized and Virtual View Management Service

This service supports the creation and population of (possibly integrated) views, using either a virtual approach, a materialized approach, or a hybrid of the two. In the case of (partially) virtual views, both the Query Result Integration Service (M.1.c.4) and the Caching Service (M.1.c.5) are used. In the case of (partially) materialized view, the Caching Service is used, possibly with incremental maintenance being supported using some of the Active Services (M.4). The Distributed Query Generation (M.1.c.3) and Query Result Integration (M.1.c.4) Services might also be used when initially creating the materialized view, or when making batch updates to it.

M.1.c.3:   Distributed Query Generation Service

This is used for programming the creation of integration bases from multiple sources, and when executing queries against integrated views that are wholly or partially virtual.

M.1.c.4:   Query Result Integration Service

Integrating the results of subqueries resulting from M.1.c.3 that are made against multiple information sources.

M.1.c.5:   Caching Service

This includes caching for the purpose of integrating the results of subqueries of a query, and caching to support a materialized (possibly integrated) view. In the latter case, automatic incremental updating might be supported using Active Services (M.4)

M.2: Query Services

These support user query specification and reformulation, and the browsing of various information sources, to access their data, metadata, constraints, knowledge, and control knowledge,

M.2.a: Schema Query Services

M.2.a.1: Browsing meta-data

Users may wish to browse schema (meta-data) information of the common schema as well as schemas of constituent information sources. This service allows users to effectively browse these schemas.

The meta-data browsing service may request services from other services such as wrapping access services to retrieve schema information from constituent information sources.

M.2.a.2: Browsing meta-knowledge

This service permits users to examine meta-knowledge, i.e., control knowledge which governs the behavior of certain system functions and services.

This service is based on the premise that the meta- knowledge will be expressed explicitly in a declarative language which is readable and understandable by humans. The meta-knowledge browsing service may make request service from Ontology and Vocabulary Transformation Services (M.1.a.1) and Control Elemental Services (M.7.c).

M.2.a.3: Formulation Services

The query formulation service supports users in the specification of complex queries against a common schema of objects and relationships. This schema may represent the integrated schema or view derived from the integrated schema.

Formulation services would have sufficient knowledge to correct common query language syntactic and semantic errors. This service uses several other services such as Browsing Services on meta-data (M.2.a.2) and data (M.2.b.1), Cost Estimation (M.6.c.2) to predict the cost of query evaluation in terms of time, space and other costs, etc.

M.2.a.4: Reformulation Services

This service is used if the query formulation and query processing services do not provide a satisfactory answer to the user's intended query.

Reformulation Services allow users to generalize queries by including broader terms, by relaxing conditions and predicates, and by allowing attribute values to be within ranges of values rather than match exact values. Conversely, queries may be specialized so as to restrict the search space and reduce the amount of data resulting from a query.

M.2.a.5: Semantic Heterogeneity Resolution

This service resolves the difference in meanings associated with objects from different information sources. The service consults the respective ontologies to resolve differences in meanings. This resolution may involve semantic transformations, generalizations, or the specification of new terms for inclusion in a common integrated ontology. This service would consult the elemental description, comparator and resolver services of M.7.

M.2.a.6: Relaxation Services

These services are used by the Reformulation Services (M.2.a.4) to relax conditions on a query so as to increase the likelihood of obtaining a meaningful answer. A query may be relaxed along various dimensions: 1) Attribute values may be specified within a range rather than a single value, 2) a term or concept may be augmented by synonymous concepts taken from the common ontology, and 3) terms may be generalized/specialized to broaden/focus the specification.

M.2.a.7: Association Services

The association service "associates" additional information to a query result that has not been specifically requested by the user, but is intended to enhance the amount of information communicated to the user.

Associated information is based on domain knowledge and the user's interest profile. Domain knowledge is represented by past cases of query processing which contain user feedback related to the relevance of the query results.

Associated information provides benefits in reducing the number of queries posed by the user, enhancing the actual information presented to the user after each query, and reducing the communication bandwidth by reducing the number of queries issued by the user.

For example, if a pilot requests the location of an airport for an emergency landing, the system might associate the runway condition and the weather at the airport.

M.2.b: Data Query Services

M.2.b.1: Browsing Data Services

M.2.b.2: Browsing Knowledge Services

M.2.c: Control Query Services

M.2.c.1: Query Planning Services

M.2.c.2: Query Optimization Services

M.2.c.3: Multiple Query Optimization

M.3: Multistate Services

These support the management of change, including schema evolution, time-stamping, versions and version management, data replication, and hypothetical access to information.

M.3.a: Schema Multistate Services

M.3.a.1: Schema Version Service
M.3.a.2: Time-Stamp Service

M.3.b: Data Multistate Services

M.3.b.1: Data Version Service
M.3.b.2: Replication Service
M.3.b.3: Hypothetical Access Service

This provides support for hypothetical access to information sources, as arises in hypothetical reasoning and simulations.

M.3.c: Control Multistate Services

M.3.c.1: Version Management Service
M.3.c.2: Replication Management Service

M.4: Active Services

These provide support for automated response to given conditions, where the conditions and the response might be specified in a relatively declarative fashion. This includes, for example, event monitoring at the request of a customer, automatic incremental updating of a materialized view within an I^3 application, or maintaining consistency between multiple information sources that hold related or overlapping information.

M.4.a: Schema Active Services

M.4.a.1: Active View Materialization Services

This service monitors the schemas which constitute a view specification for changes that occur to the underlying schemas and changes the view specification accordingly. This service would use a schema evolution service in M.3.

M.4.b: Data Active Services

This suite of services addresses the active database issues in cases in which users and applications wish to have certain events and conditions monitored in one or more databases and appropriate actions taken automatically. These actions may simply alert the user of the monitored event or condition, or they may execute a collection of rules specifying actions to update databases, materialize views, or fill subscriptions for objects.

M.4.b.1: Alerting Services
M.4.b.2: Monitoring Services
M.4.b.3: Subscription Services

M.4.c: Control Active Services

M.4.c.1: Event-Condition-Action Rule Execution Services

This supports the specification and implementation of execution models for the application of ECA rules, in the sense of active databases.

M.4.c.2: Update Triggering and Propagation Service

This suite of services control the updates between information sources when interrelated objects are updated. Primary applications include the contexts of incremental maintenance of materialized views, creating or refreshing of materialized views, and maintaining consistency information sources holding related or overlapping data.

M.4.c.3: Rule Conflict Resolution Strategies

This service provides a suite of conflict resolution options for deciding the order of rule firing in active database monitoring and execution. This service would work closely with the E-C-A Rule Execution Service (M.4.c.1) and the Update Triggering and Propagation Service (M.4.c.2).

Note that certain active database query specification languages such as Starburst have PRECEDES and FOLLOWS clauses for rules that order them with respect to firing order. OPS5-based production system models use the conflict resolution strategies inherent to these systems: specificity ordering, rule ordering, data ordering, size ordering, and recency ordering.

M.4.c.4: Workflow Services

These services provide mechanisms by which a collection of services can be coordinated in terms of temporal execution and information sharing.

An example of such a service can be found in the PACT prototype developed by Lockheed. (Cf. Cutkosky, M., R. Englemore, et al. (1993). PACT: An Experiment in Integrating Concurrent Engineering Systems. IEEE Computer. 26: 28-37.)

M.5: Explanation Services

Support for the explanation of mediation results not only to users but also to other I^3 services so that they can be made aware of the lineage of mediated results, the confidence associated with those results, and the context within which the results are applicable.

M.5.a: Schema Explanation Services

This suite of services explains the various aspects of schema integration and transformation activities, by providing the lineage of events and actions that were used to create common schema objects.

M.5.a.1: Schema Integration Explanation

This service explains the derivation of integrated schema objects, their attributes, and their relationships to other common schema objects, in terms of the underlying data elements from the constituent information sources.

M.5.a.2: Schema Transformation Explanation

This service explains the derivation of transformed schema objects, their attributes, and their relationships to other common schema objects in terms of the underlying data elements from the constituent information sources.

M.5.a.3: View Explanation

This service explains a view specification in terms of the query specification, mediated information, transformations and other possible services that are used to materialize that view.

M.5.b: Data Explanation Services

M.5.b.1: Data Format Explanation

This service provides a description of the data format for a schema attribute. It uses the services provided by the Description Service (M.7.a.1)

M.5.b.2: Data Lineage Explanation

This service traces the "lineage" of a data item, explaining all aspects of the transformations, mergings, abstractions, and versions in which it has participated. This service provides a "history" of all actions that have affected the object.

M.5.c: Control Explanation Services

M.5.c.1: Mediation Process Explanation
M.5.c.2: Coordinator Process Explanation
M.5.c.3: Wrapping Process Explanation
M.5.c.4: Inference Explanation

M.6: Resource Management Services

This collection of services allow for the planning, management, optimization and overall coordination of I^3 system services. These include the scheduling of query processing, the optimization of multiple queries, query planning and processing strategies, data movement during query processing, and the maintenance of replicated data.

M.6.a: Schema Management Services

M.6.a.1: Repository Support Services
M.6.a.2: Directory Support Services

M.6.b: Data Management Services

M.6.b.1: Ontology Support Services

M.6.c: Control Managment Services

M.6.c.1: Interface Services
M.6.c.1: (1) Facilitator Reporting Services
M.6.c.1: (2) Wrapping Calling Services
M.6.c.2: Cost Estimation

M.7: Elemental Services

This includes miscellaneous services providing basic support for most or all of the previous services, including inference, database, ontology, comparator, and resolver services.

M.7.a: Schema Elemental Services

M.7.a.1: Description Service

This service takes a term defined in one ontology and provides the context for that term in terms of its meta- data, constraints, usage, scope and applicability.

M.7.a.2: Comparator Service

M.7.a.3: Resolver Service

M.7.b: Data Elemental Services

M.7.b.1: Description Service

The Descriptor Service is used to define a term (i.e., data item or attribute) in language specified in an ontology. The term can be defined by any number of facets such as its ontological meaning, temporal aspects, or its quality (e.g., accuracy, completeness).

FIGURE 4.2-3 GOES HERE

As an example, consider the use of a descriptor service to determine the meaning of a database attribute, Blood_type , from a Hospital database. The descriptor uses a hospital records ontology as input. The result is a description of the Blood_type attribute. The structurre and semantic of the input and output must be specified by the description service and may vary depending on ontology, context and data models.

FIGURE 4.2-4 GOES HERE

M.7.b.2: Comparator Service

The comparator service is used to compare to descriptions and determine conflicts between those descriptions. For example a comparator service may be used to determine structural conflicts (i.e., address in on source is the same as city, state and zip in another), semantic differences (i.e., units, derived data definitions), data quality differences (i.e., accuracy of patients vital signs at different readings), and temporal differences (i.e., different versions of patients medical statistics). the output of the compartor service is a clear description of the conflicts that exist between the data or knowledge inputs.

FIGURE 4.2-5 GOES HERE

M.7.b.3: Resolver Service

The resolver service is used to resolve differences as presented by a comparator language specified in an ontology.

FIGURE 4.2-6 GOES HERE

M.7.c: Control Elemental Services

M.7.c.1: Inference Service
M.7.c.2: Ontology Service
M.7.c.3: Database Service

4.3) Examples

We now present two broad examples that illustrate mediation services. The first focuses on the family of Information Integration Services, and the second focuses on a subset of these, including Ontology and Context Transformation Services.

4.3.1) Example Illustrating the Information Integration Services

The Information Integration Services (M.1) allow the designer of an I^3 application to program the integration of multiple, heterogeneous information sources into one "integration base". We consider now an example of a large-grained service that uses the Information Integration Service.

In the framework of the five echelon Army trauma care example, a wounded soldier's medical records from echelon 1, 2, and 3 care must be integrated with echelon 4 and 5 records, which are the permanent, complete records. The way this is done now is completely manual, often by physically transporting each soldier's permanent records overseas (in a paper file). One could easily imagine that after a military mission, all 1, 2, and 3 echelon field care records could be merged with permanent records. This way, echelon 4 and 5 databases would not be transported overseas, and the integration could be done automatically.

The integration would most likely be done in a batch fashion, using the Information Integration Service. There would be one independent batch run for every echelon 4 or 5 database that must have echelon 1, 2, and 3 records merged into it after the mission is completed.

Given that local databases at echelons 1, 2, and 3 might well be on paper or at best stored in a local PC-based DBMS, the Schema Integration Services (M.1.a) would be used to create mappings between the local data sources and the permanent databases. The Data Integration Services (M.1.b) would be used to match up field records with individual patient records in the permanent databases (Object Correspondence -- M.1.b.3) and to create summary medical records in the permanent databases (Abstraction and Aggregation -- M.1.b.2) at echelons 4 and 5.

Various Access Policies Services (M.1.c) would be used to control the processing of these batch updates. In this case, the Materialized and Virtual View Management Service (M.1.c.2) would be used to support the construction of the integrated view, which in this case is materialized. The flow of data from the echelon 1, 2, and 3 sources to the integration bases at echelons 4 and 5 might be controlled by the Caching Service (M.1.c.5) and the Update Triggering and Propagation Service (M.4.c.2).

The Distributed Query Generation Service (M.1.c.3) and the Query Result Integration Service (M.1.c.4) would be used to program the mappings from sets of echelon 1, 2, and 3 information sources and individual echelon 4 and 5 integration bases. The Distributed Query Generation Service would also ensure that the integration base is locked, and that all local databases are appropriately locked, assuming they are electronic. (This is the reason the word "Query" is in the name of these services; they must in part ensure a distributed, heterogeneous transaction protocol.) There would be no updates to the integration base that would propagate to the local sources at echelons 1, 2, and 3, and so the Distributed Update Generation Sevice (M.1.c.1) would not be used.

As a final note, one could imagine using the Update Triggering and Propagation Service (M.4.c.2) to incrementally update the echelon 4 and 5 databases from the field, and not waiting for a batch update. This of course assumes that these databases are electronic, and it is reasonable to assume that in the future, they will be.

4.3.2) Examples Illustrating the Transformation Services

This example focuses on Disease Exposure Risk Level Assessment.

For rapid troop deployment it is necessary to understand the medical hazards that may be faced by the troops being deployed. It will be important for Echelon 1, 2 and 3 level to be prepared for the level of impact for area of deployment diseases. For example, if chicken pox was known to be common in the area of deployment it would be useful to know the aggregate number of persons at risk as determined by the individual identity of those that had history of chicken pox or had received an immunization. At Echelon 3 aggregate information will be important for assessing risk of exposure and if possible to order supplies to administer vaccine or medical supplies that may help reduce the severity of illness. At the Echelon 1 information about individual personnel will help identify how limited resources of vaccines and medical supplies can be most effectively distributed. Knowing this information will require the ability to access detailed medical records of personnel and to aggregate at the division or company level. This may require access to patient records in the field (Echelon 1, 2, and 3) as well as information that can be obtained from Echelon 4 and 5 records or home data records such as those found in hospital and health maintenance organizations.

Identification and analysis of information sources are needed to locate the information required to determine risk exposure levels for an potential or assumed disease exposure. One can imagine a scenario where an analysis at the battalion level would want to know individuals that need to be immunized or administered scarce supplies while at the division level early aggregate information would be useful in ordering supplies and to assist in developing a battle strategy based on available supplies and exposure level risks. In order to derive this information Echelon 3 level will need to access to home patient records either as stored by the military or as available through hospital or HMO information systems.

I3 Services will be essential in locating, integrating and accessing appropriate source of information to provide aggregate and specific risk exposures. Mediation Services will be used to query, integrate and reconcile information found in multiple information sources.

To begin, access to this information will require the Ontological and Vocabulary Transformation Service simply to identify the concepts in the various data stores that will be needed to make the assessment. For example, one data store might identify prior diseases as text in Patient_History while others represent explicitly as fields such as ChickenPox (i.e., with encoded values showing prior exposure or level of vaccination). Schema Transformation Services would be used to map to common concepts and structure. Once common concepts have been determined Query Services can be used to create multidatabase requests. Trying to merge such information will present an object identity problem as Echelon 3 request might use military identification (i.e., Company A) for personnel whereas hospital records might use individual name or hospital identification numbers. The Object Correspondence Service can help map aggregate data objects into the individual objects and then map those to object identifications in other data stores.

After identifying common concepts and locating common objects additional integration services are needed to assist the Query

Services. At the data-level, context transformation may be needed to consolidate and compare data that has very different assumptions or meaning. Often encoded information will have to be resolved to some common terminology. For example, database may include coded data to identify which strains of vaccine have been previously administered. For this purpose, Elemental Data Services can be used to construct the needed Context Transformation Services. First, a Description Service is used to describe a data item in ontological terms. Then a Comparator Service can be used to determine the difference between this description and one required by the receiver application. Finally, a Resolver Service can be used to transformation the information into the receiver's context. A wiring diagram for the construction of context mediation services is shown in the figure below (** to be constructed). Here the context mediator is used to resolve differences between data identifying that an individual has received a vaccine.

The example shows that once schema translation issues have been resolved, data translation issues must also be addressed. In the Hospital database the value indicating the vaccine has an associated context where the type of vaccine (i.e., vaccine_type) is a function of the date of administration. In the echelon 3 level application it has the threat of a given disease has been determined the desired vaccine has been determined. The Context Mediation Service then determines the conflict between the explicit vaccine information request with the information as found in the Hospital Database.

At Echelon 3 level the ability to access all information may be too costly or time consuming. Query Services will assist in accessing multiple databases and combing the answers as required. The Abstraction/Summarization Service can be used to aggregate a sampling of information that can give a reasonable estimate of the true risk level.

5) WRAPPING SERVICES

5.1) Overview and Crisp Definition

Wrapping services act as transducers between legacy information sources and higher level services. Wrapping services have two primary purposes:

- To allow various coordination and mediation services to manipulate legacy information sources that were not specifically designed to be compatible with these coordination and mediation services.
- To be as re-usable as possible; wrapping services should, whenever possible, provide common interfaces (to information sources) that follow widely accepted standards. These might include using a standard SQL as a query language interface to relational DMBS's, or using CORBA as an object communication protocol. This will allow individual "wrapped" information sources to be accessible to a wide class of coordination and mediation services.

Wrapping services must handle a broad range of incompatibilities, since various information sources may be of widely different natures. This involves transforming communication interfaces, data structures, and program semantics. In essence, a wrapping service takes an information source and modifies its interface, its data, and/or its behavior so that it is accessible by the outside world, and it attempts to do this by using accepted standards.

The true goal is to standardize the process of "wrapping" information sources in order to eliminate redundant effort and to incrementally build a library of reusable, "wrapped" information sources. This should also facilitate the development of new "wrapped" information sources, since the wrapping services in the library will be built using an organized set of sub-services (see immediately below); therefore, the process for implementing a wrapping service tool (and not just the tool itself) should be reusable.

5.2) High-Level Wrapping Services

Below is a hierarchy of the primary wrapping services and their sub-services.

W.1: Communication Wrapping Service
- W.1.a: Calling Interface Service
  (for transforming the calling interface between two programs, written in either the same or different languages)
- W.1.b: Event Management Service

(for managing event traffic between an information source and an incompatible higher level service)

W.1.c: Method and Function Calling Service
(for transforming method and function calls between an information source and an incompatible higher level service)

W.2: Data Restructuring Wrapping Service

W.2.a: Data Format Restructuring Service
(for translating between different data formats at the machine or file level)

W.2.b: Meta Data Restructuring Service
(for transforming meta data formats at the machine or file level)

W.3: Behavioral Transformation Wrapping Service

W.3.a: Application Program Modification Service
(for altering the semantics of a program, such as adding precision to a calculation - this often involves "wrapping" internal modules within an application program and exposing their interfaces, so that selected modules can be modified and/or reused independently.)

W.3.b: Protocol Modification Service
(for altering the protocols of a system, such as concurrency protocols - this often involves "wrapping" an internal component of an otherwise closed system, so that an internal protocol can be exposed.)

W.3.c: Data Language Transformation Service
(for translating between different data manipulation languages, such as incompatible SQL's)

One significant issue is the separation between Mediation and Wrapping Services. It is not always clear where to assign an individual function. For example, 3.a might well be considered more of a Mediation Service by some individuals. The exact separation is still a research issue, but the heuristic used in this document is that if the transformation is relatively small grained and low-level from a semantic point of view, and if it produces a wrapped information source that is likely to be widely used, than the service belongs in the Wrapping Services hierarchy.

For example, if a service takes a program that supplies the square roots of integers and allows the precision to be adjusted, then this is a wrapping, and not a mediation service. On the other hand, if a service takes a set of data values and produces a complex aggregation for export out of the information source, this is probably an Abstraction and Aggregation mediator service. As a final example, theD Data Format Restructuring Service might transform form entries into records or tuples (see below), but the Object Correspondence mediation service would be responsible for matching up tuples from one information source with objects from another information source; again, a mediation service performs tasks that are semantically rich.

This brings up a related issue. There is nothing wrong with a mediation or coordination service directly manipulating an information source if no specific wrapping service is required.

5.3) Example

Recall the batch update application for the Army Trauma Care example described in Subsection 4.3.1. In this example, echelon 1, 2, and 3 level medical records are merged in a batch mode with echelon 4, and 5 permanent medical records. It is assumed that many of the lower echelon records are stored on paper. Tools supporting Wrapping Services would be used for the following purposes:

1. Data Format Restructuring Service (W.2.a):

   Forms based records must be keyed in (in batch) into a local (echelon 1, 2, or 3) computer using some sort of interactive, forms-based clerical interface. This may or may not be done in the field. These form entries are then transformed by the Data Format Restructuring service into records (or relational tuples), ready to be integrated into the larger database by the Mediation Service.

   It may well be the case that some of these form entries are already in machine readable format, probably in a small computer. In this case, they do not need to be keyed in, but they still must be transformed into records.

2. Data Language Transformation Service (W.3.c):

   Most likely, the various echelon 4 and 5 information sources are not independent. From time to time, it will be necessary to coordinate the data in multiple sources because they involve common patients. In this case, the Data Language Transformation service would probably be used to translate between incompatible SQL's (for example, between Oracle and Sybase).

3. Protocol Modification Service (W.3.b):

   In cases where echelon 4 and 5 databases must be coordinated, a distributed, heterogeneous locking protocol will be needed in order to ensure integrity during the coordination phase. The Protocol Modification Service would be used to expose the lock managers and the lock tables of individual echelon 4 and 5 databases. This would allow the construction of locks that span multiple systems. If for example, tuples from a relational database were being coordinated with objects from an object-oriented DBMS (because the tuples and objects represent the same patient), both the tuples and the objects would have to be locked simultaneously. This would demand "wrapping" and exposing internal components of the individual DMBS's.

AFTERMATTER

APPENDIX A: Diagrams Illustrating the Reference Architecture

A.1) table on the various major I^3 services and their brief definitions

** not prepared yet

A.2) tree showing type/subtype relationships between
major services, one for each of the 3 layers, (e.g.,
the translation meditation service has subtypes restructuring
and abstraction)

** not prepared yet (early drafts of these were developed
at Stanford Workshop)

Figure A.2-1
Figure A.2-2
Figure A.2-3

A.3) wiring diagrams for selected services

** some are embedded into the body of the document; others
are not prepared yet.

APPENDIX B:
</XMP>

<H1>I^3 Vocabulary: </H1>

(For the latest version, click

here)

<XMP>

APPENDIX C: Further Details Concerning the Five Echelons of Army Trauma Care

Echelon 1:
Care is provided by designated individuals or elements organic to combat units, and elements of hte area support medical battalion. Major emphasis is placed on those measures necessary ot stabilize the patient and allow for evacuation to the next level of support. Sub elements of this level include

(a) Self-aid and buddy aid.
(b) Combat Lifesaver
(c) Combat Medic
(d) Treatment squads (Battalion Aid Station).

Echelon 2: At this level care is rendered at the clearing station (division or corps). Here the casualty is examined and his wounds and general status are evaluated to degtermine his treatment and evacuation precedences. Emergiency medical treatment including beginning recuscitation, is continued, and, if necessary additional emergiency measures are instituted, but they do not go beyond the measures dictated by the immediate necessities. The division clearing station has a whole blood capacity and limited x-ray and ambulatory services. Patients who can RTD (return to duty) witin 1 to 3 days are held for treatment. These functions are performed typically by company-size medical units organic to brigades, divisions and ASMBs. Medical Company is an example of this type of a unit.

Echelon 3: At this level, the patient is treated in an medical treatment facility staffed and equipped to provide resucitation, initial wound surgery, and post-operative treatment. Patient who are nontransportable due to their medical condition may receive immediate surgical care in a hospital in the division rear area (MASH unit). Those patients whose condition permits further movement back are evacuated. Those patients who are expected to RTD are regulated to a facility that has the capability for reconditioning and rehabilitation.

Echelon 4: at this level of medical care, the patient is treated in a hospital staffed and equipped for general and specialized medical and surgical care and reconditioning and rehabilitation for return to duty.

Echelon 5: Continental United States Base Support. This also includes reconditioning, rehab and long term support for those who are not expected to RTD. The CONUS based support represent the most definitive medical care available within the Health Services Support system.

APPENDIX D: Further Details Concerning OMG/OSA

Section 2.3.1 gave a brief introduction to OMG/OSA
and its relationship to the I^3 Reference Architecture.
Sections D.1 and D.2 here give more detail in support
of Subsections 2.3.1.1 and 2.3.1.2.

The appendix also contains a variety of other information
concerning the OMG effort, including:

-- D.3: The benefits of OMG/I^3 cooperation
-- D.4: Organization of OMG
-- D.5: Levels of OMG membership
-- D.6: OMG Architectural Principles
-- D.7: Availability of OMG specifications
-- D.8: Template for Describing OMG Object Services and Common Facilities
-- D.9: Example of a completed OMG CFA service descriptor

This appendix contains more detail concerning the OMG/OSA.
The first two sections give more detail

xx it is currently somewhat fragmented and not well synchronized
with Section 2.3.1 on OMG/OSA. it is undergoing some
revisions

D.1 More detail on the four aspects of the OMG framework

Section 2.3.1.1 gave a brief overview of the OMG architecture. This
section presents more detail concerning that architecture, along with
enumerations of services that have been developed or are under
development. This architecture is expanding rapidly, and so this
description must be viewed as a dated snapshot.

As noted in Section 2.3.1.1, the {\italicize OMG Object Management
Architecture Guide}, completed in 1990, serves as an overall
architecture and roadmap for OMG standards development. It describes
OMG's standard reference architecture, shown in Figure 2.3.1-1. The
architecture consists of components, interfaces, and protocols which
decompose an object system into four areas, classified as follows:

The OMG Common Object Request Broker Architecture (CORBA 1.1 and 1.2)
specification provides a "message passing bus" for transporting requests
and responses in a distributed environment. CORBA also defines IDL

(Interface Description Language), a language neutral object model with separately specified mappings to C++ and Smalltalk (so far) as well as a specification for a standard Interface Repository for IDL meta data. CORBA 2.0 extends CORBA to require that all CORBA implementations support Internet TCP/IP and optionally DCE as transport protocols. The OMG ORB Task Force has issued a COM/CORBA Interworking RFP in two parts, one addressing the current Microsoft COM and one focusing on Microsoft's future distributed COM (DCOM, alias Cairo). Work is progressing on defining interfaces to enable interoperability between different ORB implementations, so that, for example, a SUN ORB would seamlessly interoperate with an IBM or DEC ORB. The ORBTF is considering a possible RFP on CORBA Application Portability for later in 1995. Other future possible work includes templates for IDL, replication service, synchronization service, group broadcast dispatch, asynchronous dispatch, messaging queuing, additional language bindings, monitoring, and other services.

The OMG Object Services Architecture defines a collection of "basic object services" that most applications need. Detailed specifications for many of these services are complete, including OMG Object Services Task Force (OSTF) RFP1 services (Life Cycle, Events, Naming, Persistence) and RFP2 services (Externalization, Concurrency, Transactions, Relationships). Responses to RFP3 services (security services) have been received and a merging is expected by August 1995. OSTF RFP4 covers Object Queries, Properties, and Licensing services. The Query service, based on an object SQL, has been adopted (?); the other two specifications are expected in May 1995. RFP5, just issued (?), will cover Change Management (versions, configurations, dependencies/transformations), Collections Library, and Trading services. OSTF is considering issuing an RFP for DBMS, OMG's first composite service, built from base OMG services, facilities, and CORBA. Potential future services include schema evolution, archive, backup/restore, implementation repository, installation and activation, startup, and threads.

The OMG Common Facilities Architecture provides a second collection of object services (called common facilities) that provide general purpose capabilities, useful in many applications. Common Facilities Task Force (CFTF) RFP1 covers compound documents presentation and interchange. RFP2 will cover Time Operations and Internationalization. CFTF RFP3 will cover Data Interchange and Automation. CFTF RFP4 will cover Scripting, Rule Management, and an Agent Facility. A CFTF RFI on Repository will determine which services a repository needs and which new services are still needed. Similar to the OSTF RFP on DBMS, this RFI will help define how to compose OMG services into a useful composite service. Future common facilities work is expected in the areas of messaging, task management, work flow, help, mediators, negotiators, various electronic commerce services, distributed simulation, and GIS. There are horizontal Common Facilities used by all applications (most of those above) and vertical Common Facilities that support specific application domains such as manufacturing, healthcare, finance, and telecommunications. So far, there are no specifications for services or domain models in these industry vertical areas. It may make sense for OMG to split out these latter areas into a new Domain Objects Task Force.

Application Objects are specific to particular end-user applications and are not subject to standardization.

In the OMA Reference Model, the main purpose of the categorization of objects into Object Services, Common Facilities, and Application Objects is to set the standardization strategy for the OMG. All services have IDL interfaces and communicate with each other via CORBA. Object Services include more basic object interfaces; Common Facilities provide standardized interfaces to common application services; and Application Objects reflect the need for independently-developed application interfaces for which the OMG will never develop standards.

An OMG ''Object Service'' (used generically here to include basic object services and common facilities) defines the interfaces and sequencing semantics to support building well-formed applications in a distributed object environment.

In non-object software systems, a system's Application Program Interface (API) often is defined by a monolithic interface. The OMG Object Services API is modular; particular objects may use a few or many Object Services. By being object-oriented, the OMG Object Services API is extensible, customizable, and subsettable; applications only need to use services they require.

The operations provided by Object Services are made available through the IDL Interface Definition Language (as defined in the OMG CORBA specification) or through proposed extensions to IDL compatible with the OMG Object Model.

While OMG requires an IDL interface for each object service, it is worth noting that since OMG adopts existing technology, it may be the case that implementations of object services may not themselves be object-oriented and that, in addition to an IDL interface, non-object-oriented interfaces may continue to be supported for compatibility with an existing product's API or with non-OMG standards. Such interfaces will not however be part of OMG specifications. Also note that objects do not have to use the implementation of basic operations provided by Object Services nor do objects have to provide all basic operations. For example, an object may provide its own data storage or meta data access.

D.2 More detail on the relationship of OMG and I^3 Reference Architectures

This section gives more detail concerning how I^3 might use the OGM standards when implementing I^3 services, and on the overlap between the services under development by both communities.

I^3 can immediately depend on some existing ORB related specifications: OMG IDL, which provides a language neutral object model that the I^3 community can adopt; CORBA, which provides distribution; and the IDL Interface Repository, which provides a standard representation for accessing IDL meta data, including message signatures. CORBA implementations on the market provide these services as well as IDL to C++ mappings. OMG provides a Program Invocation Service (though there may not be a formal spec, all ORBs support this capability) for registering and invoking new services. The wakeup daemon aspect of the I^3 Program Invocation Service, not yet specified in OMG, can be accomodated by OMG's Event Service and programmable message dispatch capability (also analogous to the NIIIP VE Monitor).

In terms of the I^3 architecture, the CORBA Naming Service and Interface Repository seem to provide I^3 White and I^3 Yellow Pages respectively and CORBA itself may provide a basic I^3 Context-Based Routing/Broker and 3 Program Invocation Service. If the CORBA Interface Repository does not represent all information needed by the White and Yellow Pages, then extensions of the CORBA Interface Repository specification may be needed. Extensibility of the CORBA Interface Repository is an OMG requirement since new services may add new meta data. But we need to understand how to do this modularly in an OMG standard way. I^3 could help here.

Some of the most direct I^3 connections are with OMG services that are identified and scheduled for RFP but for which no specifications yet exist. The OMG Change Management Service will support a version service, configuration service, and dependency/transformation management service. There may be a close relationship between the proposed I^3 configuration manager and I^3 configuration designer services. The OMG Dependency/Transformation Service is intended to maintain consistency between two representations. A similar control capability is needed for the I^3 Transformation Service. The main job of the I^3 Transformation Service seems to be to convert data from one representation to another. The as yet unspecified OMG Data Interchange Service does this as well. The I^3 service appears to work on ontologies represented in KIF; the OMG service is generic. Almost certainly it will end up providing a general framework for transformational mappings with potentially hundreds of subclasses that internalize and externalize the many data interchange formats into/out of IDL. Similarly, the I^3 Comparator Service will need to provide a generic framework for comparing representations, which will be specialized by domain-specific comparators that are representation dependent. The OMG change management service will depend on this capability as well.

OMG CFTF is considering an RFI for a general "repository." Industry has spent a lot of effort trying to define this term and its responsibilities. ANSI X3H4 was long involved in this effort. The I^3 community is using this term as well. It is important to be clear what is meant: the set of services that the repository provides (e.g., an engine) or a set of representations that can be stored in a DBMS, KBMS, or other engine and that that can be accessed, queried, transformed, and otherwise operated on.

The OMG Object Query Service specification is most like some of the proposed higher level I^3 services in providing a way to decompose a query into a collection of subqueries that may or may not access heterogeneous data sources. Its optimizer, encapsulated and hidden inside the current query specification, may internally use a variation of the proposed I^3 Scheduling and Sequencing Service. It will be interesting to see what this I^3 service does that goes well beyond what the Query Optimizer needs.

OMG has identified the need for a Rules Service (CFTF RFP September 1995). This appears close to the intent to the I^3 Inference Service.

I^3 Smart Yellow Pages, Facilitator, and Matching Services all appear similar in that they use the Inference/Rules system as well as Interface and domain repository to locate, transform, or compare information. The OMG Query Service could be viewed as a simpler version of some of these services that does not use all the power of a rules engine. The OMG Trader Service (OSTF RFP March 1995) is most like the I^3 Smart Yellow Pages Service in providing a smart path to identifying services.

It is not clear inside OMG how scripting, rules, and any ACL might interrelate. They all appear to provide a means of specifying behavior, which OOPLs like C++ and Smalltalk also provide. Perhaps they will be viewed as a compatible family of behavioral specification languages. If so, there are additional specification language that are candidates for stating integrity constraints and invariant conditions on OMG IDL objects and perhaps I^3 agents.

No OMG services so far proposed provide a framework for mediation, facilitation, or negotiation. I^3 could add these to Common Facilities.

In addition to the above, additional benefits may come from the I^3 community only indirectly related to the I^3 Reference Architecture: an IDL to Common Lisp mappings is needed. Also, IDL could benefit from Common Lisp's meta object protocol, which would involve working with OMG's Object Model Task Force.

All of the above services appear generic and it would be valuable if one framework could be made to support so many services and capabilities uniformly. This is directly in line with the overall OMG goal is of providing a rich family of services that makes it easier to build complex applications.

OMG is evolving fairly rapidly but it is important to note that there is typically a one year lag from RFP to adopted specification and another one year lag to commercial implementation. The bad news is that not all OMG service specifications or implementations are immediately available to the ARPA I^3 program. The good news is that there is still time to affect OMG specifications and develop candidate reference implementations that can be of use to the OMG community in the near term.

D.3 On the benefits of cooperation between OMG and I^3

If the OMG and ARPA I^3 reference architectures can be made interconsistent, the following benefits will accrue:

- I^3 results will have an immediate and visible technology transfer route into broader industry.
- I^3 can build on and extend the OMG architecture, avoiding reinventing infrastructure others will provide.
- since I^3 results will help solve classes of semantic interoperability problems, they will make the OMG architecture substantially more useful in providing ways to connect legacy and heterogeneous applications and systems.
- I^3 ideas will be critically reviewed by a variety of communities, including other groups with agent and knowledge-based approaches.

While it is not expected that all useful results of the I^3 program are candidates for transfer to OMG, substantial synergy is possible if planned for in advance.

D.4 Organization of OMG

OMG was founded in 1989 and today has over 500 member organizations. OMG meets bi-monthly usually with every other meeting outside the U.S. A Board of Directors approves final decisions of the Technical Committee (TC), which includes all OMG member organizations. Voting rights depend on subscription level but non-voters can wield considerable technical influence in committees. Most work occurs in OMG Task Forces, Special Interest Groups, or Subcommittees, which all report to the TC. Task Forces typically issue Requests for Information (RFI) to help define a part of OMG's overall Object Management Architecture, then define an architecture and roadmap and issue a series of Requests for Proposals (RFPs) to industry for interfaces to "services" (components) that populate the architecture. RFP respondents provide candidate interface specifications and promise conformant commercially available implementations. From RFP issue to technology adoption is typically one year; commercial implementations are typically available a year after that. There are currently four TFs: Object Request Broker, Object Model, Object Services, Common Facilities (mirroring the OMG architecture). There are currently several active SIGs: Financial, Manufacturing, Healthcare, Business Objects, and End User (mirroring OMG's potential for providing industry-specific object standards). There are currently two standing subcommittees: Policies and Procedures and Liaison, the latter concerned with insuring that OMG builds and maintains close liaisons with other industry groups and standards development organizations.

At the March 1995 OMG meeting, the OMG leadership communicated their intention to reorganize the structure of OMG to better architect and integrate the many and growing collection of OMG components. A new subcommittee will make organizational recommendations to the Board of Directors sometime during the Summer. The organization will potentially consolidate or restructure efforts in both the Task Force infrastructure areas as well as the domain specific Special Interest Groups.

D.5 Levels of OMG membership

Corporate members join OMG with voting rights on the Technical Committee (one vote per organization). Corporate members can sponsor technology specifications for adoption by the Technical Committee. Members may attend all OMG technical meetings, sub-committees, and task forces as well as interact closely with OMG staff. They can also serve on advisory and business committees in setting OMG policies.

Associate/University members are non-voting members. Associate members can attend all technical, sub-committee and task force meetings of the OMG, and their voices will be heard.

End User/Government Agency members are organizations that will internally implement and deploy object technology. End User/Government Agency members help the OMG keep on course with real world requirements. They are represented as a Special Interest Group to the Technical Committee and feed requirements into the adoption process. This is a critical working group and carries a collective vote in the Technical Committee. They can participate in all Task Forces.

Subscribing members receive the OMG newsletter and have access to all technical OMG documents. This resource of information is unequaled in the world, as a central source on object technology.

OMG is a non-profit organization. OMG membership dues are tax exempt.

For information on membership contact renee@omg.org.

D.6 OMG Architectural Principles

The OMG architecture is distributed, open, and non proprietary. It is being built on existing and emerging industry standards, exploits object-oriented technologies, and creates a "plug and play" environment to embrace both new and legacy applications. OMG object services obey a collection of architectural principles. Respondents to RFPs must indicate how their proposed specifications satisfy these architectural requirements, helping to insure a consistent overall architecture.

- OMG object service interfaces shall have precise descriptions. Specifications shall be complete. There must be no "magic" required.

- OMG service interfaces shall be object-oriented and shall be described in OMG's IDL.

- Any part of a service that is viewable as an object shall be viewed that way and shall be given an IDL interface description.

- Any requirements that a service places on general objects shall be defined in IDL. For example, a security service might define what a "securable" object is in IDL.

- Operation sequencing shall be specified where applicable. Behavior and sequencing of operations shall be part of the specification of a service.

- Service specifications shall not contain implementation descriptions. Maximum implementation flexibility should be preserved.

- Data and process management services must support cooperative protocols that facilitate logically distributed control in scalable, heterogeneous environments. Data and process managers that support local centralized environments must bridge to the logically distributed environment by support of the cooperative protocols in addition to supporting their local centralized protocol.

- Independence and modularity of services: It should be possible to separately specify and implement each service. Monolithic systems must be viewable as a collection of independent services.

- Services shall be factored for reuse. Factoring shall minimize duplication of functionality (the Bauhaus principle).

- To facilitate modularity, independence, and reuse, functionality should belong to the most appropriate service (not bundled).

- Legacy systems shall be accessible via IDL wrappers.

- Services shall interoperate when there are dependencies.

- Interfaces and behaviors should be specified sufficiently well so that implementations can be replaced and the parts will still work together

- Different services should be able to work together.

- Services should be extensible through an iterative process. It should be clear how extensions can be defined (e.g., via multiple inheritance, delegation, etc.).

- It should be possible to define new services without having to re-design a system that uses existing services.

- Meta data, repository, or knowledge base services should not be required in all implementations. For instance, meta data needed by some service may be available from repositories or knowledge bases or may be added at compile time or manually.

- Integrity, reliability, and safety of services: safeguards should be provided to guard against corrupting services and the objects they manage.

- Services should be designed to accommodate portability of implementations across a wide range of platforms

- Interfaces to services should be designed with performance trade-offs in mind. Services and compositions of services must provide adequate performance.

- Scalability: services should be specified with the goal that there may be many implementations that are optimized for different environments. IDL and a language mapping can make remote services look like local object invocations. Most services have a possible implementation across a spectrum from a shared library to a remote server. The user of a service should not have to care which variety is being used, unless interested.

D.7 Availability of OMG specifications

The following OMG specifications are available from OMG or are on OMG roadmaps. In the cases of already adopted specifications, short descriptions are provided here but there are substantial specifications (30pp to 1000pp) available from OMG.

OMG specifications are copyrighted by OMG. OMG processes exist to revise specifications. OMG is interested in constructive comment on existing or new specifications.

The following documents are only available from OMG in hardcopy format.

```
   Object Management Architecture Guide . . . . . . . . . . . . . $ 50
   Common Object Request Broker Architecture 1.2 . . . . . . . . . NC
   Common Object Request Broker Architecture 2.0[Draft]. . . . . NC
   Common Object Services Specification, Volume 1. . . . . . . . $ 50
   Object Analysis and Design: Description of Methods  . . . . . $ 50
   Object Analysis and Design: Comparison of Methods . . . . . . $ 50
   OO Methodologies, Databases and Languages Report. . . . . . . $725
```

Contact

Object Management Group
Framingham Corporate Center
492 Old Connecticut Path
Framingham, MA 01701--4568
Telephone: +1-508-820-4300
Facsimile: +1-508-820-4303

OMG's WWW homepage is under construction: http://ruby.omg.org/index.html.

OMG provides a document server. Send email to server@omg.org with a body

```
  help
  index
  get docs/doclist.txt
  get docs/yy-mm-nn.[ps txt rtf]   where yy-mm-nn is an OMG document number
```

to receive OMG documents.

The following is a list of key OMG architectural documents and specifications.

```
OMG Object Request Task Force
=============================
 CORBA 1.1               Defines IDL and CORBA's distributed dispatch.
 CORBA 1.2               CORBA Revisions.
 IDL C++ Mapping         Standard mapping for IDL to C++.
 IDL Smalltalk Mapping   Standard mapping for IDL to Smalltalk
 IDL Interface Repository Standard definition of IDL meta data.  Does not
```

```
                        address storage, query, versioning, ... of this
                        information since other services do that.
 CORBA 2.0              Requires TCP/IP interoperability as common transport
                        between CORBAs; makes DCE optional.

OMG OBJECT MODEL TASK FORCE
===========================

 OMG Object Model Reference Model
                        Defines abstract core object model plus the
                        notion of components (additions to the core) and
                        profiles (standard families of additions).

OMG OBJECT SERVICES TASK FORCE
==============================

 OMG Object Services Architecture
 OMG Object Services Roadmap

OSTF RFP1 (COSS-1 -- adopted)
 Lifecycle              Supports object create, delete, copy, and equivalence.
 Event Notification     Supports notification of events to interested objects.
 Persistence            Supports persistence of an object independent of both
                        the lifetime of the client applications that access
                        the object and of the implementation that realizes the
                        object's methods.
 Naming                 Supports mapping between names and objects.

OSTF RFP1 (COSS-2 -- adopted)
 Externalization        Supports object externalization and internalization.
 Relationships          Supports associations between two or more objects.
 Concurrency Control    Supports concurrent access to one or more objects by
                        one or more objects.
 Transactions           Supports atomic execution of one or more operations.

OSTF RFP1 (COSS-3 -- 3 specifications being merged -- adoption expected August 1995)
 Security               Supports authentication and authorization, providing
                        discretionary access control on objects, not
                        precluding mandatory access control.  Does not address
                        encryption.

OSTF RFP1 (COSS-4 -- Query service adopted; Properties expected in June)
 Query                  Supports object SQL operations on sets and collections
                        that return sets and collections.  Interface does not
                        address indexing.
 Properties             Provides dynamic named attributes associated with an
                        object.
 Licensing              Supports license management.
```

```
OSTF RFP1 (COSS-5 -- RFP issued in March 1995)
 Collections             Supports a Collections Class Library
 Change Management       Supports identification and consistent evolution of
                         objects including version and configuration management
                         as well as dependency maintenance among related objects.
 Trading                 Supports matching of provided services to the service needs
                         of a client object.

OMG Common Facilities Task Force
================================

CFTF RFP1 (RFP issued in October 1994; submissions reviewed in March 1995)
 Compound Documents     Covers Presentation and Interchange

CFTF RFP2 (RFP planned for March 1995)
 Time Operations        Supports synchronization of clocks in a distributed system.
 Internationalization

CFTF RFI on Repository  Aim is to determine which services a repository needs
                        and which new services are still needed.

CFTF RFP3 (RFP planned for September 1995)
 Data Interchange       Supports the exchange of some or all of an object's visible
                        state between two objects.
 Automation

CFTF RFP4 (RFP planned for September 1995)
 Scripting              Supports scripting languages.
 Rule Management        Supports integrity constraints and
                        event-condition-action rules.
 Agent Facility         Supports agents technology.
```

D.8 Template for Describing OMG Object Services and Common Facilities

To make I^3 specifications compatible with OMG specifications, one prerequisite is to match OMG documentation formatting guidelines. OMG uses an OMG standard template and process to define services.

In the OMG OSA and CFA reference architectures, detailed descriptions of individual Object Services and Common Facilities are provided to give general guidance on how future work may proceed. Individual object services are described using the template below:

- Service Description and Requirements

Prescriptively describes an object service; delimits the scope of the service. The description is written in the form of a collection of requirements ("shall provide") and goals ("should address").

- Related Standards and References

  References that should be useful to submitters in understanding the problem space. Submitters should describe whether and how their proposals relate to existing, relevant industry standards and other work.

- Relationship to other Object Services, CORBA, and the OMG Object Model

  Possible interactions with (or relationship to or dependencies on) other Object Services, Common Facilities, CORBA, and the OMG Object Model.

- Technical Issues

  Technical issues that the Object Service Architecture addresses and open technical issues not yet addressed.

An example is given below.

D.9 Example of a completed OMG CFA service descriptor

The following is an example of an OMG Common Facilities Architecture service descriptor for one of the OMG services, the Rules Management Facility. This descriptor is part of the CFA reference architecture. This description will be included in CFTF RFP4 to be issued in September 1994. (The numbering is from the CFA reference architecture.)

Rule Management Facility

E.3.1 Service Description and Requirements

The Rule Management Facility provides for declarative event-condition-action rule specification and processing. Rule Management further involves the acquisition, management, and execution of Rules.

The Rule Management Facility shall provide:

- a rules specification language

- a rule execution engine.

The Rule Management Facility should address:

- uses of rules (e.g., dbms integrity, constraints, workflow triggers, kbms queries)

- forward chaining, backward chaining, certainty factors

- active rules, deductive rules, integrity constraints

- how rules relate to IDL inheritance hierarchies (e.g., right-hand-side of rules as message bodies, rule inheritance)

- rule maintenance (e.g., insertion, deletion semantics)

- external representations and interchange format for rules

- federation of rule management facilities (e.g., composing rules services, accommodating heterogeneity and distribution in communicating rules services)

E.3.2 Related Standards and References

- X3J21 Specification Languages Z and VDM

- X3T2 Knowledge Interchange Format (KIF), conceptual graphs, and conceptual schema modelling facilities.

- ARPA Knowledge Sharing program including KIF, KQML, Agents, Mediators, Negotiation

- Prolog, OPS5

- KBMS systems like LOOM, Neuron Data

- Active DBMS systems like Postgres, ODE, OSAM* KBMS

E.3.3 Relationship to Other Common Facilities, OMG Object Services, CORBA, and the OMG Object Model

The rule service depends on:

- OMG IDL (mandatory)

- COSS-1 Event Service (TBD) -- for active rule driven systems

- CORBA, COSS Persistence Service, COSS Security Service, others (optional) -- for distribution, persistence, security, versioning, meta data repository, etc.

E.3.4 Technical Issues

- granularity of rule sharing across heterogeneous rule-based systems

- rulebase validation (e.g., detecting cyclic rules, subsumption, inconsistency, redundancy, tradeoffs)

- rulebase evolution

- interaction of rules and the Object Transaction Service

- relationship of rules and the OMG OSTF Object Query Service

- rule scope
    - in organizational hierarchies
    - active in a temporal interval or spatial region

APPENDIX E: Further Details Concerning NIIIP

This appendix gives more detail concerning the NIIIP consortium. Section E.1 includes some general background, and Section E.2 presents a brief summary of the 13 major NIIIP components, and their relationship to the high-level I^3 services.

E.1) General Background

The NIIIP consortium is a $60M program, including substantial funding from the government's Technology Reinvestment Program. Nineteen organizations are involved: CAD Framework Initiative, Digital Equipment Corp., Enterprise Integration Technologies, General Dynamic/Electric Boat Division, IBM (Manufacturing Industry, Networking System, Software Solutions, Research), International TechneGroup Inc., Lockheed (LASC), Magnovox Electronic Systems, NIST, RPI, STEP Tools Inc., Taligent, Texas Instruments, UES Inc., and the University of Florida.

NIIIP has identified certain key technologies that industrial virtual enterprises will need to depend on. At present, these technologies are not well integrated. These technologies include: Internet WWW, OMG, STEP Product Modeling, Work Flow, KBMS, and agent technology. A key NIIIP technical goal is to provide technical solutions integrating these technologies and then work with owner communities to get the integrating solutions broadly adopted.

Information about NIIIP is available from http://www.niiip.org.

E.2) The 13 major components of MIIIP

We now list the 13 high-level components of the NIIIP Architecture, and their relationship to the I^3 services. For each NIIIP component we first give a brief description, and then an indented paragraph describing the relationship to I^3.

1. Desktop

Provides representation of the VE to people; an administrative interface to VE components; is the target of NIIIP requests to people; supports multiple VEs per person; supports multiple sessions per VE.

> This focuses on user interface issues, and is out of the scope of the I^3 Program

2. Transport

Separates data transport from transport implementation; transparently supports files, objects, messages via CORBA, TCP/IP, FTP, HTTP, S-HTTP, DFS, etc.; facilitates caching from within the component as well as through OMG's COSS (Common Object Services Specification); facilitates high availability through its replication services; facilitates use of multiple security modes from within the transport component

> Issues such as network communication protocols are viewed to be out of the scope of the I^3 program. On the other hand, the caching and replication services of the NIIIP Transport component are closely related to the Mediation Services of Caching (M.1.c.5) and Update Triggering and Propagation Service (M.4.c.2).

3. Agent

Provides for creation and management of an agent system; supports decision making over owned resources and/or use of resources owned by others relative to their price, cost, time of delivery, and quality of result; implements S-HTTP for other components; has "preferences" and "goals"; monitors project planning "timeline" data.

> In its current form the I^3 Reference Architecture does not advocate any implementation paradigm for tools supporting I^3 services (see Subsection 2.2.3). However, as discussed in Section 2.3.3, the agent paradigm has many useful features in connection with implementing such tools. Furthermore, a few of the I^3 Coordination Services have aspects that incorporate sophisticated decision making.

4. STEP

Provides the ISO STEP Level 3 services; three dozen data/model interfaces; supports a high-level repository interfaces over a distributed physical databases.

> As noted earlier, the ISO STEP standard provides common terminlogy and compatible representations for a variety of domains. In this respect, NIIIP is favoring the developing of standards, whereas I^3 favors the use of Mediator Services (specifically, M.1 of Section 4) to perform transformations between differing representational paradigms

5. Data

Provides a uniform interface to shared databases; ANSI-SQL, ODBC, OMG-93; provides uniform interface to Change Management Systems (configuration and version control); provides a uniform interface to data and schema translations, including data compression and encryption.

> These functionalities are mostly supported by the I^3 Data Restructuring Wrapping Service (W.2) although I^3 is less focused on mapping to a single uniform interface for a given functionality. Data and schema translation is supported in I^3 by the Data Integration (M.1.b) and View Transformation (M.1.a.2), although a uniform interface is not required

6. Workflow

Models organization, roles, activities; enacts workflow activities and simulations; distributes work across VE member boundaries; provides activity templates for reusable activities; collects metrics; enhances workflow functionality with frames and triggers; provides an interface to tools and application data; defines dependencies between objects.

> While there is an I^3 Workflow Services (M.4.c.4 in Section 4), this application area is not a focus of the I^3 Reference Architecture in its current form.

7. Session

Manages the context and state of the VE system and environment; provides access to data, tool, task, and other shared computing resources; provides for modeling project, task; provides for synchronous and asynchronous events; acts as an agent for the user-directed events; provides homogeneous representation of heterogeneous resources to the user; supports scalable, distributed network resource environments; logs session data; post-processes session logs.

> Speaking broadly, managing the context and state of an I^3 application is handled by various Coordination Services (???), and event traffic is handed by the Active Services (M.4). Providing a homogenous representations of heterogenous information is performed primarily by the Information Integration Service (M.1) for semantic heterogeneity, and by various Wrapping Services for heterogeneity of a more syntactic or systems nature.

8. Tool

Provides access to data encapsulated in a tool; prescribes marshalling requirements, movement of data to function and function to data; provides transform services; engages in "tool-talking".

Some aspects of this are not addressed within I^3. Providing access to data encapsulated in a tool can be supported by some of the I^3 Wrapping Services, including in particular the Application Program Modification Service (W.3.a)

9. Mediator

Specifies the naming, structural and semantic relationships among data of heterogeneous systems using a mediation language; compiles specifications and translates them into object classes which contain data to carry out the mediation process; integrates mediated classes into the VE global schema; causes automatic triggering of mediation events; accepts mediation queries and automates data conversions; accommodates mediation rules for abstraction, summarization, and translation.

The general function of mediation is perrfomed by I^3 Mediation Services. In an I^3 applications, compiling specifications and creating tools that execute Mediation Services might be performed by hand, or might be performed by Coordination Services.

10. Negotiator

Supports a trading environment; supports an extended four step object request protocol (ask, agree, deliver and accept); resolves differences between contending agents through analysis of their "preference functions"; supports open and/or devious behavior; recognizes 2-party and n-party negotiation; various "styles" of negotiation are supported; implements ideas from Harvard Business School (Prof Raiffa).

Many of these functionalities can be supported by the Facilitator Service, although negotiation is not at present a primary focus of the I^3 Reference Architecture.

11. VE KBMS

Provides buildtime facilities to create a VE; manages VE knowledge base in support of the other 12 VE components; provides query language; query optimization; natural language query support; generates one approach to the VE Request Monitor; provides the canonical form of the NIIIP rule processing component.

In an I^3 application, management of the services and information resources is generally accomplished by various Coordination

Services, including primarily Management Services (C.2) and/or Facilitation Services (C.3.b). Support for queries against multiple information sources and query optimization is provided primarily by Access Policies Services (M.1.c); mapping information sources to a uniform query langauge is supported by the Data Language Tranformation Service (W.3.c). Rule processing in I^3 is handled primarily by Active Services (M.4).

12. Internet Tools

Provide access to Internet via gateway services; provide a NIIIP gateway for Internet Clients; provide NIIIP VE members Internet Multimedia Conferencing; provides a "home page" for the VE.

The I^3 Reference Architecture assumes the availabity of Intenet tools.

13. VE Monitor

Trap object requests (method calls); mediate ill-formed requests; route request to VE components to enforces VE rules, security requirements, data translation, mediation, agent negotiation, etc.; manage the cascade of "hidden" VE operations.

In the I^3 Reference Architecture these services will generally be supported by the Management Services (C.2) and the Active Services (M.4). (Both of these might in turn might use other services for specific tasks.)

APPENDIX F: Further Details Concerning the Paradigm of Agents

This appendix gives a broad overview of an architecture based on the agent paradigm. Subsection E.1 summarizes the substrate on which the architecture is based, viz. distributed object management. Subsection E.2 gives a graphical presentation of an organization of agents based on federations. Subsection E.3 describes some characteristics of the Agent Communication Language (ACL), and Subsection E.4 gives a characterization of agents.

The terminology and definitions used here apply only to this Appendix, and may not coincide with terminology in the document proper.

E.1) Distributed Object Management Substrate

An OBJECT is a software component that communicates with its peers by message passing. This excludes, in particular, all forms of implicit communication via environmental side effects.

Objects can be as simple as subroutine groups; but typically they are larger entities with some sort of persistent control (e.g. distinct control threads within a single address space, distinct processes on a single machine, or separate processes on different machines).

A wide variety of distributed object management frameworks have been proposed and implemented, e.g. Corba, OpenDoc, and OLE. All provide reliable message passing between (possibly) distributed objects, and most provide some form of location-transparency (i.e. communication by name rather than location). The I^3 architecture depends on these features but, in all other respects, is independent of any particular distributed object feature.

Note that AGENTS are objects of a special sort. Like any other object, an agent provides a message-based interface independent of its internal data structures and algorithms. The primary difference lies in the language of the interface. In communication between objects, the meaning of a message can vary from one object to another. By contrast, agents use a common language with an agent-independent semantics, viz, the agent communication language described below.

Note also that some tools supporting I^3 services may well be written in the form of objects that communicate in languages other than the universal language. It is expected that such objects will be represented in an I^3 system by representative agents capable of using the universal language to communicate with other agents in the system.

E.2) Federation Architecture

Figure ARCH (E.2-1) presents a diagrammatic representation of a system based on agents. Application agents (user interface programs, information repositories, and other information processing programs) are ‘‘plugged’’ into the system, which is responsible for ensuring their interoperation. As suggested by the diagram, in this pure view, application agents do not communicate directly with each other. Instead, they communicate only with facilitators; and facilitators communicate for them, with the assistance of directory agents and mediators.

<FIGURE ARCH showing agents, mediators, repositories, facilitators connected by ACL linkages>
[Figure E.2-1]

In this architecture, agents use the agent communication language to convey their needs and abilities to their directories. In addition to this metalevel information, they also send application-level information and requests to their facilitators and accept application-level information and requests in return. Facilitators call on mediators and use the information stored in directories to transform these application-level messages and route them to the appropriate places.

The agents in such a configuration form, in effect, a ‘‘federation’’ in which they surrender their autonomy to their facilitators and the facilitators take the responsibility for fulfilling their needs. Hence the name FEDERATION ARCHITECTURE.

It is noteworthy that this view is strictly a logical view. In fact, application programs are expected to be distributed on multiple machines, as are system components. Figure PHYSICAL (E.2-2) shows how such a system might look in practice.

<FIGURE PHYSICAL showing physical view with multiple facilitators>
[Figure E.2-2]

It is also worthy of note that the architecture described above is, in a sense, an ideal. In operation, it is expected that these sharp lines of communication are likely to be adjusted, as suggested in Figure VARIATION (E.2-3).

<FIGURE VARIATION -- direct connections to mediators, other languages -- created manually or through matchmaking services of facilitators>
[Figure E.2-3]

These variations are likely to occur for two reasons. First, facilitators can set up connections between components, after which the components communicate directly with each other, possibly in specialized languages. Second, system implementors can configure systems in various ways, on the basis of prior knowledge about the capabilities and needs of the relevant components (thus solving the problem of heterogeneity) and on the basis of assumptions about their stability (thus dealing with the problem of dynamics).

We note in particular that there are emerging commercial products, such as UniSQL and O2's DBAccess, that support limited forms of heterogeneous database integration, and essentially provide a form of a federated architecture. These tools might well be worth using to instantiate I^3 applications. These tools allow data stored in muiltiple, heterogeneous commercial DBMS's to be viewed through a common object model and interface.

E.3) Agent Communication Language (ACL)

Communication language standards facilitate the creation of interoperable software by decoupling implementation from interface. So long as programs abide by the details of the standards, it does not matter how they are implemented. Today, standards exist for a wide variety of domains. For example, electronic mail programs from different vendors manage to interoperate through the use of mail standards like SMTP. Disparate graphics programs interoperate using standard formats like GIF and JPEG. Text formatting programs and printers interoperate using languages like PostScript.

Unfortunately, problems arise when it becomes necessary for programs that use one language to interoperate with programs that use a different language. To begin with, there can be INCONSISTENCIES in the use of syntax or vocabulary. One program may use a word or expression to mean one thing while another program uses the same word or expression to mean something entirely different. At the same time, there can be INCOMPATIBILITIES. Different programs may use different words or expressions to say the same thing.

The agent paradigm attacks these problems by prescribing that components provide support for a universal communication language, one in which inconsistencies and arbitrary notational variations are eliminated. Note that this does NOT mean that all communication must occur within this language. IN PARTICULAR, COMPONENTS MAY ALSO USE OTHER, SPECIALIZED LANGUAGES FOR COMMUNICATION WITH SIMILAR COMPONENTS. The universal language is to ensure that components share AT LEAST one language.

There are two popular approaches to the design of a universal language

-- the procedural approach and the declarative approach.

The procedural approach is based on the idea that communication can be best modelled as the exchange of procedural directives. Scripting languages (such as TCL, Apple Events, and Telescript) are based on this approach. They are both simple and powerful. They allow programs to transmit not only individual commands but entire programs, thus implementing delayed or persistent goals of various sorts. They are also (usually) directly and efficiently executable.

Unfortunately, there are disadvantages to purely procedural languages. For one, devising procedures sometimes requires information about the recipent that may not be available to the sender. Secondly, procedures are unidirectional. Much information that agents must share should be usable in both directions -- to compute quantity A from quantity B at one time and to compute quantity B from quantity A at another. Most significantly, scripts are difficult to merge. This is no problem so long as all communication is one-on-one. However, things become more difficult when an agent receives multiple scripts from multiple agents that must be run simultaneously and may interfere with each other. Merging procedural information is much more difficult than merging declarative specifications or mixed mode information (like condition-action rules).

In contrast with this procedural approach, the declarative approach to language design is based on the idea that communication can be best modelled as the exchange of declarative statements (definitions, assumptions, and the like). To be maximally useful, a declarative language must be sufficiently expressive to communicate information of widely varying sorts (including procedures). At the same time, the language must be reasonably compact; it must ensure that communication is possible without excessive growth over specialized languages.

As of this writing, it is not clear which of these two approaches will succeed. There is an evident need for declarativeness in the long run. However, scripting languages are likely to be popular in the short run because of their familiarity; and so the ultimate agent communication language may end up looking more like a scripting language than a declarative language.

Of special note in this regard is ACL, a coordinated suite of languages developed by researchers in the ARPA Knowledge Sharing Effort. ACL can best be thought of as consisting of three parts -- its vocabulary, an ''inner language'' called KIF (short for Knowledge Interchange Format), and an ''outer'' language called KQML (short for Knowledge Query and Manipulation Language). An ACL message is a KQML expression in which the ''arguments'' are terms or sentences in KIF formed from words in the ACL vocabulary. ACL as a whole is notable because it offers the familiarity of a scripting language while providing the expressiveness of a declarative language.

Note that, although the components of ACL were designed with this joint use in mind, each of the components can be used independently of the others; and all have been used in this way. Furthermore, KIF and KQML are the subjects of separate standardization efforts in ANSI and OMG.

In communicating information it is essential to preserve semantics. Sharing information across a community demands agreement about the meaning of symbols, e.g., the meaning of the vocabulary for objects, functions, and relations in ACL. In a small setting it may be possible to mandate a single vocabulary for all agents to use in communication. However, in a large setting, this will be impossible. More than likely, different communities In such an environment will assign different meanings to the same symbols. Such an environment will require support for a collection of vocabularies or ontologies. To disambiguate the meanings of symbols, all communication between agents will specify the ontology used. In order to support communication between agents using different ontologies, the system also needs the definitions of articulation axioms to relate the symbols in one vocabulary to the symbols in another vocabulary.

The discussion about multiple ontologies and articulation axioms is a topic in itself, and beyond the scope of this section.

E.4) Constraints on Agent Behavior

The criterion for agenthood is a behavioral one. An entity is a software agent if and only if it communicates correctly in a suitable agent communication language. This means that the entity must be able to read and write well-formed messages, and it means that the entity must abide by the behavioral constraints implicit in the meanings of those messages.

The specific constraints associated with a message derive from the content of that message and general principles of agent behavior. For example, there is veracity (an agent must tell the truth), autonomy (an agent may not constrain another agent to perform a service unless the other agent has advertised its willingness to accept such a request), commitment (if an agent advertises a willingness to perform a service, then it is obliged to perform that service when asked to do so), and so forth.

From a theoretical perspective, it is interesting to note that all of these principles can be derived from the single principle of veracity. In other words, if all agents are constrained to tell the truth, then autonomy, commitment, etc. all follow. To many people, the principle of veracity sounds too strong; but it is not difficult to achieve. An agent can always state its own inputs, outputs, and definitions with confidence; and it can nest conjectures inside of statements about its ‘‘beliefs’’. Unfortunately, a full account of this issue is beyond the scope of this paper; and, interesting as it may be theoretically, it has only indirect practical value.

For our purposes here, it is sufficient to say that the use of an agent communication language brings with it behavioral constraints. However, this leaves opens a wide range of possibilities. At one extreme, we can imagine ‘‘perfect’’ agents that retain all of the information they receive and act in accordance with the logical consequences of this information. At the other extreme, we can imagine simple agents, like calculators, that answer arithmetic problems and ignore everything else. More powerful agents utilize a larger portion of the language; less powerful agents use a smaller subset. All are agents, so long as they use the language correctly.

E.5) Creating Agents

Given a clear statement of the language and the behavioral principles that agents must satisfy, it is straightforward to write programs that behave correctly. But what about all of the programs that have already been written, our so-called ‘‘legacy’’ software? Are there any standard techniques for converting such programs into software agents? In work thus far, a number of different approaches have been taken. See figure AGENTS (E.5-1).

<FIGURE AGENTS>
[E.5-1]

One approach (the leftmost diagram in figure AGENTS (E.5-1)) is to implement a transducer that mediates between an existing program and other agents. The transducer accepts messages from other agents, translates them into the program’s native communication protocol, and passes those messages to the program. It accepts the program’s responses, translates into the agent communication language, and sends the resulting messages on to other agents.

This approach has the advantage that it requires no knowledge of the program other than its communication behavior. It is, therefore, especially useful for situations in which the code for the program is unavailable or too delicate to modify.

This approach also works for other types of resources, such as files and people. It is a simple matter to write a program to read or modify an existing file with a specialized format and thereby provide access to that file via the agent communication language. Similarly, it is possible to provide a graphical user interface for a person that allows that person to interact with the system in a specialized graphical language, which is then converted into the agent communication language, and vice versa.

A second approach to dealing with legacy software (the middle diagram in figure AGENTS (E.5-1)) is to implement a wrapper, i.e. inject code into a program to allow it to communicate in the agent communication language. The wrapper can directly examine the data structures of the program and can modify those data structures. Furthermore, it may be possible to inject calls out of the program so that it can take advantage of externally available information and services.

This approach has the advantage of greater efficiency than the transduction approach, since there is less serial communication. It also works for cases where there is no interprocess communication ability in the original program. However, it requires that the code for the program be available.

We note that there has been extensive research on developing wrapper-generating tools; and within I^3 we have some experience in this realm. In particular, the interoperability tool Amalgame, which was developed for ARPA, might be useful for supporting the encapsulation and re-use of databases and complex application software that runs on top of many large database applications. It will be critical that I^3 applications allow the partial wrapping of application programs, so that portions of large systems can be isolated and re-used in isolation.

The third and most drastic approach to dealing with legacy software (the rightmost diagram in figure AGENTS (E.5-1)) is to rewrite the original program. The advantage of this approach is that it may be possible to enhance its efficiency or capability beyond what would be possible in either the transduction or wrapping approaches.

The best examples of this approach come from the engineering domain. Many automated design programs work to completion before communicating with other programs. For example, the output of a logic synthesis program is passed as input to a printed circuit board layout and routing program; its output is passed to an assembly planning program; and so forth. Recent work in concurrent engineering suggests that there is much advantage to be gained by writing programs that communicate partial results in the course of their activity and that accept partial results and feedback from other programs. By communicating a partial result and getting early feedback, a program can save work on what may turn out to be an unworkable alternative.