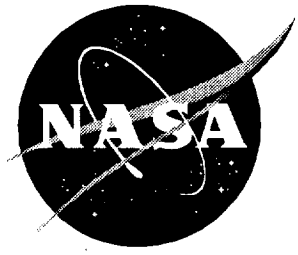
NASA/CR-1998-207679



Aviation System Analysis Capability Executive Assistant Design

Eileen Roberts, James A. Villani, Mohammed Osman, David Godso, Brent King, and Michael Ricciardi
Logistics Management Institute, McLean, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS2-14361

May 1998

Available from the following:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

Abstract

In this technical document, we describe the design developed for the Aviation System Analysis Capability (ASAC) Executive Assistant (EA) Proof of Concept (POC). We describe the genesis and role of the ASAC system, discuss the objectives of the ASAC system and provide an overview of components and models within the ASAC system, and describe the design process and the results of the ASAC EA POC system design. We also describe the evaluation process and results for applicable COTS software. The document has six chapters, a bibliography, three appendices and one attachment.

Contents

FIGURES

TABLES

Chapter 1
Introduction

NASA's Role in Promoting Aviation Technology

The United States has long been the world's leader in aviation technology for both civil and military aircraft. During the past several decades, U.S. firms have transformed this position of technological leadership into a thriving industry with large domestic and international sales of aircraft and related products.

Despite the industry's historic record of success, the difficult business environment of the past several years has stimulated concerns about whether the U.S. aeronautics industry will maintain its worldwide leadership position. Increased competition, both technological and financial, from European and other non-U.S. aircraft manufacturers has reduced the global market share of U.S. producers of large civil transport aircraft and cut the number of U.S. airframe manufacturers to only one. Order cancellations and stretch-outs of deliveries by airlines, forthcoming noise abatement requirements, and environmental concerns create additional challenges for U.S. producers and purchasers of aircraft.

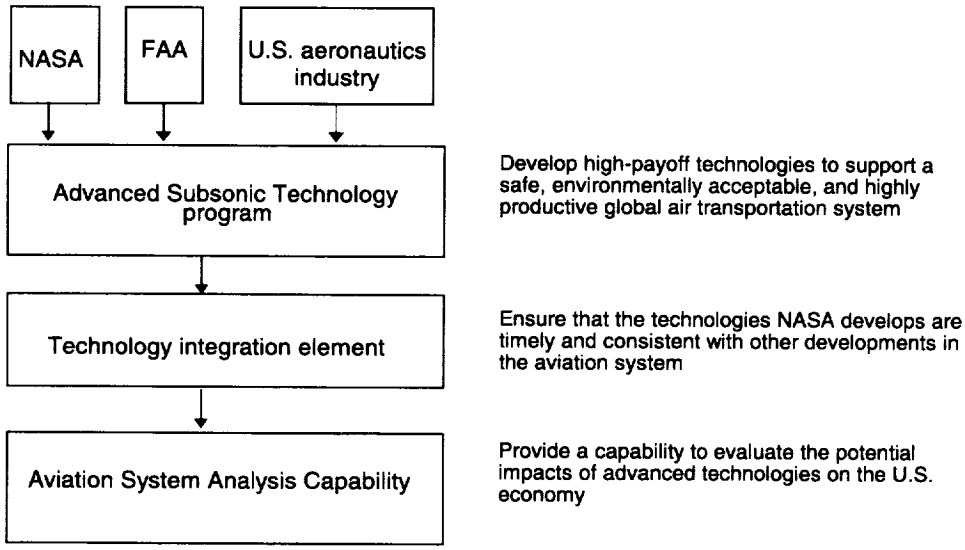
The primary role of the National Aeronautics and Space Administration (NASA) in supporting civil aviation is to develop technologies that improve the overall performance of the integrated air transportation system, making air travel safer and more efficient, as well as contributing to the economic welfare of the United States. NASA conducts much of the basic and early applied research that creates the advanced technology introduced into the air transportation system. Through its technology research program, NASA aims to maintain and improve the leadership role in aviation technology and air transportation held by the United States for the last half century.

The principal NASA program supporting subsonic transportation is the Advanced Subsonic Technology (AST) program, managed by the Subsonic Transportation Division, Office of Aeronautics, NASA Headquarters. In cooperation with the Federal Aviation Administration (FAA) and the U.S. aeronautics industry, the AST program develops high-payoff technologies that support the development of a safe, environmentally acceptable, and highly productive global air transportation system. NASA measures the long-term success of its AST program by how well it contributes to an increased market share for U.S. civil aircraft and aircraft component producers and the increased effectiveness and capacity of the national air transportation system.

NASA's Research Objective

To meet its objective of assisting the U.S. aviation industry with the technological challenges of the future, NASA must identify research areas that have the greatest potential for improving the operation of the air transportation system. Therefore, NASA seeks to develop the ability to evaluate the potential impact of various advanced technologies. By thoroughly understanding the economic impact of advanced aviation technologies and by evaluating the use of new technologies in the integrated aviation system, NASA aims to balance its aeronautical research program and help speed the introduction of high-leverage technologies. Figure 1-1 illustrates NASA's research objective.

Figure 1-1. NASA's Research Objective



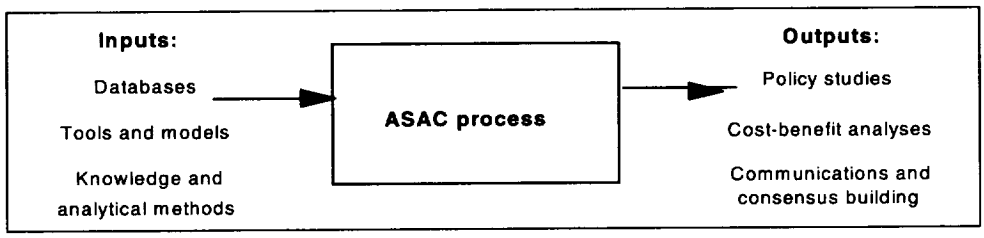
Genesis of the Aviation System Analysis Capability (ASAC)

Technology integration is the element of the AST program designed to ensure that the technologies NASA develops are timely and consistent with other developments in the aviation system. Developing an Aviation System Analysis Capability (ASAC) is one of the objectives of the technology integration element. With this analytical capability, NASA and other organizations in the aviation community can better evaluate the potential economic impacts of advanced technologies.

ASAC is envisioned primarily as a process for understanding and evaluating the impact of advanced aviation technologies on the U.S. economy. ASAC consists of a diverse collection of models, databases, analysts, and individuals from the public and private sectors brought together to work on issues of common interest to

organizations within the aviation community. ASAC will also be a resource available to those same organizations to perform analyses; provide information; and assist scientists, engineers, analysts, and program managers in their daily work. ASAC will provide this assistance through information system resources, models, and analytical expertise, and conducting and organizing large-scale studies of the aviation system and advanced technologies. Figure 1-2 displays this concept.

Figure 1-2. ASAC Process



GOALS OF THE ASAC PROJECT: IDENTIFYING AND EVALUATING PROMISING TECHNOLOGIES

Developing credible evaluations of the economic and technological impact of advanced aviation technologies on the integrated aviation system is the principal objective of ASAC. These evaluations will then be used to help NASA program managers select the most beneficial mix of technologies for NASA investment, both in broad areas, such as propulsion or navigation systems, and in more specific projects within the broader categories. Generally, engineering analyses of this kind require multidisciplinary expertise, use several models of different components and technologies, and consider multiple economic outcomes and technological alternatives. These types of analyses are most effective if they include information and inputs from organizations and analysts from different parts of the aviation community. In this way, the studies incorporate the expertise of people around the United States and build acceptance from the start of the research effort.

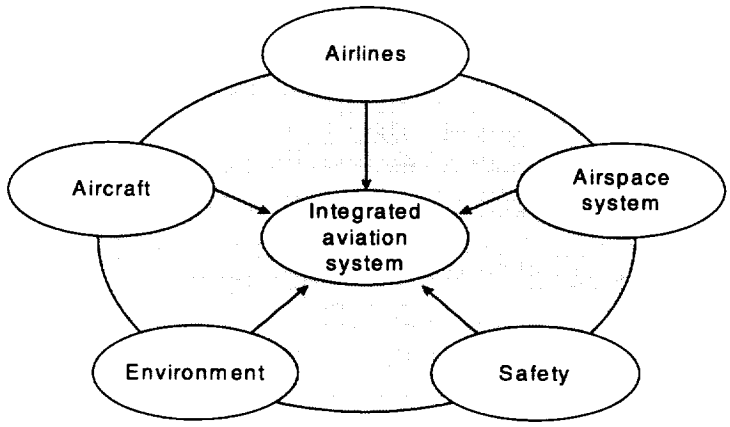
In addition to identifying broad directions for investments in technology, the program must also help researchers at NASA and elsewhere evaluate the economic potential of alternative technologies and systems. By better informing engineers about potential markets for technologies and data on how the current system works, ASAC will help NASA engineers incorporate their customers' needs more easily into their routine work. These types of problems most likely involve investigating technical designs for specific aircraft or subsystems that can readily replace existing equipment without requiring significant changes to other aviation components. With such information, researchers could more easily evaluate the utility of alternative designs and quickly estimate the value of their design concepts. Analysts from industry, government, and universities would also use ASAC in this way.

APPROACH TO ANALYZING THE INTEGRATED AVIATION SYSTEM

The most useful aviation technologies are not necessarily the most technically advanced. Rather, NASA and industry must invest in the technologies that have the most promising payoffs—those that clearly demonstrate a capacity for economically viable performance enhancements—from the perspective of those organizations that will purchase and operate the technologies.

Because new aviation technologies are introduced into a complex system, the potential impact of any proposed technology must be analyzed from a system-wide perspective. Otherwise, the potential impact may be overestimated or underestimated because of the unexamined interdependencies with other elements of the aviation system. Figure 1-3 shows the components of the integrated aviation system.

Figure 1-3. Components of the Integrated Aviation System



In summary, with the ASAC, users can develop credible evaluations of the economic and technological impact of advanced aviation technologies on all components of the integrated aviation system.

DOCUMENT OVERVIEW

This technical document a describes the system design of the Aviation System Analysis Capability (ASAC) Executive Assistant (EA). The document builds upon the work presented in the National Aeronautics and Space Administrator Contractor Report #201681, *ASAC Executive Assistant Architecture Description*

Summary, Eileen Roberts and James A. Villani, April 1997, and it is composed of the following chapters:

- Chapter 1—Introduction
- Chapter 2—Components of the Aviation System Analysis Capability
- Chapter 3—ASAC Models
- Chapter 4—Design Methodology
- Chapter 5—ASAC EA Detailed Design
- Chapter 6—Conclusion.

In Chapter 1, the genesis and role of the ASAC system is described. We discuss the objectives of the ASAC system and provide an overview of components and models within the ASAC system.

The Design Methodology chapter discusses the choice made for an architecture methodology, the Domain-Specific Software Architecture (DSSA), and the DSSA approach to developing a system design.

The next chapter, ASAC EA Detailed Design, describes the design development process and includes the ASAC EA system design. We address each DSSA design stages 4 and 5. DSSA stages 1 through 3 and partial stage 4 are detailed in the *ASAC Executive Assistant Architecture Description Summary* referenced above.

- DSSA Stage 4—Develop Domain Models
- DSSA Stage 5—Identify Reusable Artifacts.

This document has a reference, bibliography, three appendices, and two attachments:

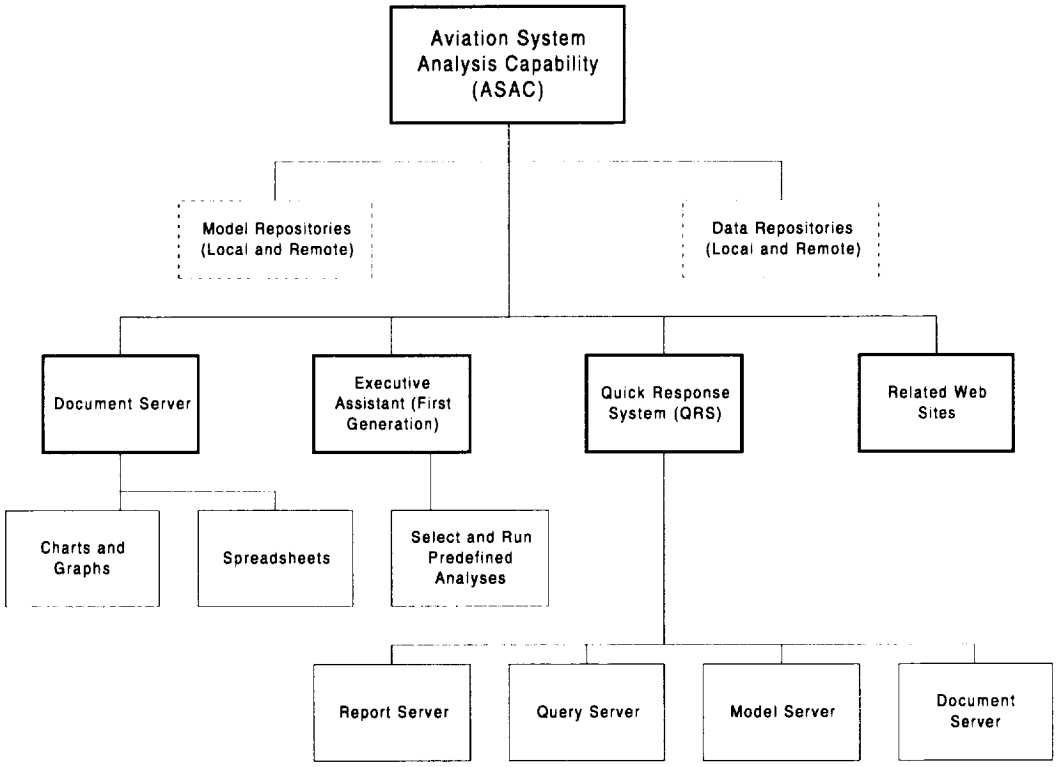
- Appendix A—Acronyms
- Appendix B—Domain Dictionary
- Appendix C—Message Broker Evaluation Supporting Documentation
- Attachment A— CORBA ORB Vendor Questionnaire Responses.

Chapter 2
Components of the ASAC

OVERVIEW

ASAC is a diverse collection of models, databases, analysts, and individuals from the public and private sectors brought together to work on the issues of common interest to organizations within the aviation community. Figure 2-1 shows the major system components of ASAC.

Figure 2-1. ASAC System Components



Some ASAC system components exist; others are under development. Two ASAC components, Related Web Sites and the Document Server are available to the general public. All other ASAC components are available on a restricted basis. The following sections provide a brief description of the ASAC Executive Assistant as it exists today. Information about the other ASAC components can be found in National Aeronautics and Space Administrator Contractor Report

#201680, *Aviation System Analysis Capability Quick Response System Report for Fiscal Year 1997*, Eileen Roberts, James A. Villani and Paul Ritter, April 1997.

ASAC EXECUTIVE ASSISTANT (EA)

With the ASAC EA, researchers at NASA and elsewhere can quickly evaluate the economic potential of alternative technologies and systems. By providing inputs to and linking the many models and data that the ASAC system will comprise, the EA will provide an intelligent interface with which the user can perform detailed analyses. Definition of the ASAC Executive Assistant design is the focus of this document.

Table 2-1 outlines the proposed development schedule for the EA.

Table 2-1. Proposed Development Schedule for the ASAC EA

Item	Year	Status
Define ASAC EA requirements	1995	Complete
Define the ASAC EA	1996	Complete
Develop the ASAC EA architecture	1996	Complete
Develop the Model Integration Prototype (First Generation ASAC)	1996–1997	Complete
Design the ASAC EA Proof of Concept	1997	Complete
Develop an ASAC EA Proof of Concept	1997–1998	Ongoing
Design, develop, and deploy the ASAC EA	1998–1999	—

Chapter 3
ASAC Models

The ASAC Model Integration Prototype (First Generation ASAC) demonstrates integration of six First Generation ASAC models. This prototype was fielded in March 1997, and is the first step in providing a robust, fully functional, ASAC Executive Assistant.

The ASAC Model Integration Prototype (First Generation ASAC) comprises a subset of the complete ASAC model network. NASA and others use it to perform selected economic analysis of aircraft technology and air traffic management improvements.

The ASAC Model Integration Prototype (First Generation ASAC) is available to authorized ASAC users (password protected). Users employ a World Wide Web (WWW) browser to access the system.

At present, six models are in the ASAC Model Repositories. Four additional models will be added to the ASAC Model Repositories shortly. The models are listed in Table 3-1. New models will be added to the repositories as they are developed.

Table 3-1. Contents of ASAC Model Repositories

Model	Operating system	Comment
Existing models		
ASAC Air Carrier Investment Model	Windows and Macintosh (Excel, Version 5.0), will be HP-UX 10.20	Available as a standalone model, will be available via a WWW interface
ASAC Air Carrier Network Cost Model	HP-UX 10.20	Available via a WWW interface
ASAC Airport Capacity Model—Detroit	HP-UX 10.20	Available via a WWW interface
ASAC Airport Delay Model—Detroit	HP-UX 10.20	Available via a WWW interface
ASAC Flight Segment Cost Model (Cost Translator)	HP-UX 10.20	Available via a WWW interface
ASAC Flight Segment Cost Model (Mission Generator)	HP-UX 10.20	Available via a WWW interface

Table 3-1. Contents of ASAC Model Repositories (Continued)

Model	Operating system	Comment
FY97 Models		
Aircraft/ATC Functional Analysis Model	HP-UX 10.20	Available as a standalone model
ASAC Airport Capacity Model—Atlanta	HP-UX 10.20	Available via a WWW interface
ASAC Airport Capacity Model—Dallas	HP-UX 10.20	Available via a WWW interface
ASAC Airport Capacity Model—Los Angeles	HP-UX 10.20	Available via a WWW interface
ASAC Airport Delay Model—Atlanta	HP-UX 10.20	Available via a WWW interface
ASAC Airport Delay Model—Dallas	HP-UX 10.20	Available via a WWW interface
ASAC Airport Delay Model—Los Angeles	HP-UX 10.20	Available via a WWW interface
ASAC Noise Impact Model	Windows NT Server 4.0	Available via a WWW interface

SCHEMATIC OF ASAC MODELS AND ANALYSIS CHAINS

ASAC models are grouped into the following four analytical areas:

- 1.0 Aircraft and System Technologies
- 2.0 FAA Air Traffic Management
- 3.0 Environment
- 4.0 General Aviation.

Each model has a unique number. The number designates the model's analytical area, e.g., all model numbers that begin with a *2* belong to the FAA Air Traffic Management analytical area. The number also designates a model's position in a logical stream. For example, a stream might comprise the following models:

2.3 ASAC Airport Capacity Model →

2.3.2 ASAC Airport Delay Model →

2.3.2.1 ASAC Flight Segment Cost Model—Cost Translator.

Model links for each of the four analytical areas are shown in Figures 3-1 through 3-4. Squares represent models that belong to the analytical area named in the figure title; and circles represent models that belong to a different analytical area.

Figure 3-1. Aircraft and System Technologies

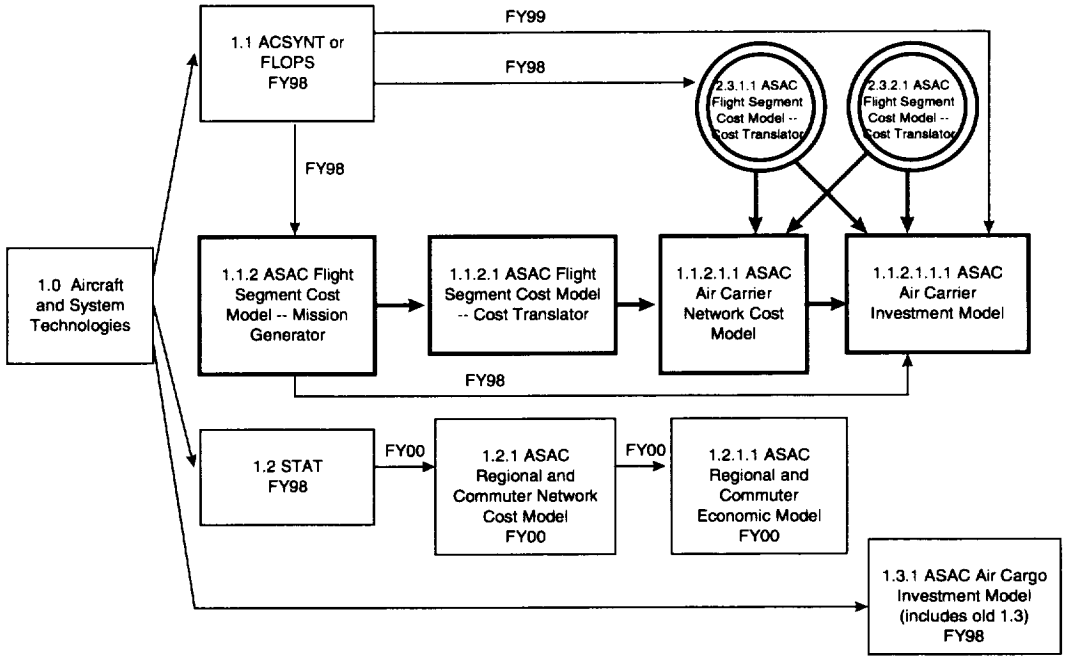


Figure 3-2. FAA Air Traffic Management

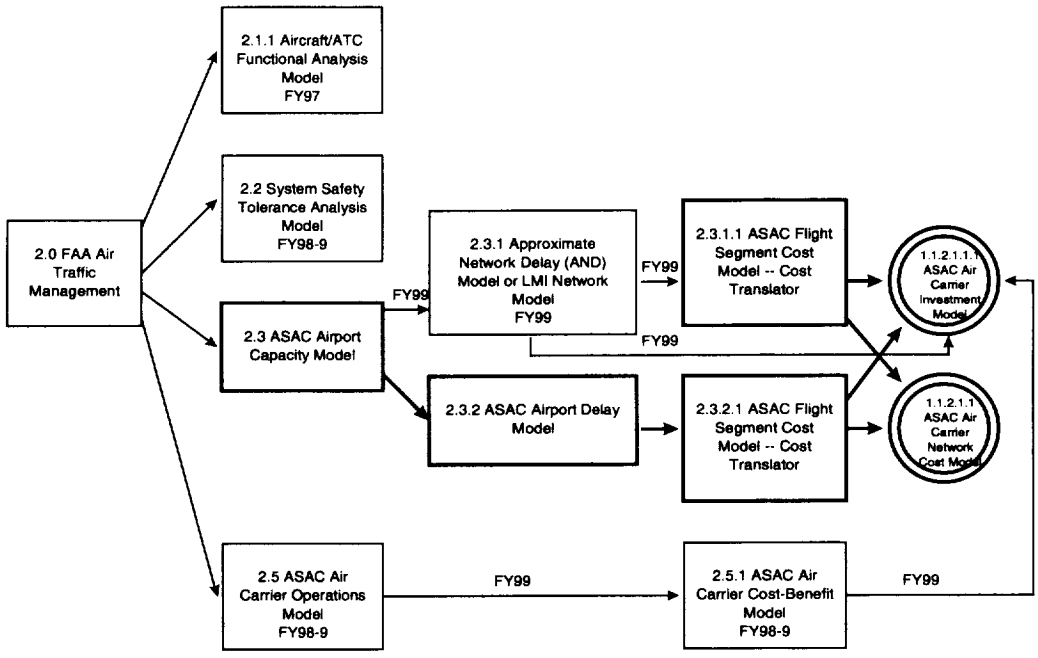


Figure 3-3. Environment

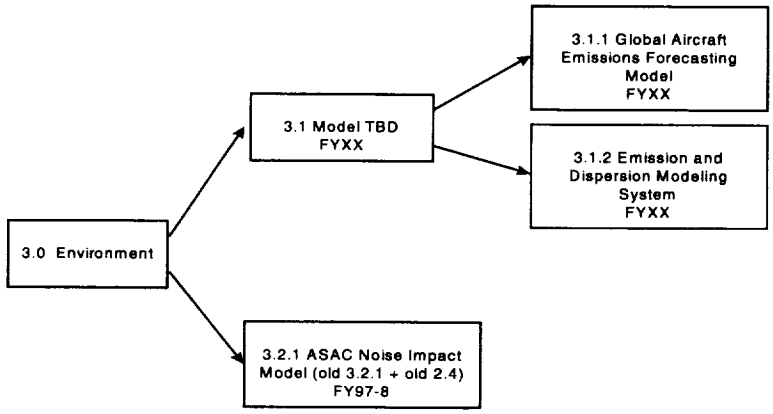
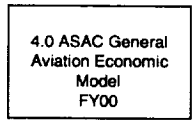


Figure 3-4. General Aviation



Analyses Using ASAC Models

The above represented models can be used either alone or in combination to analyze specific AST program elements. Table 3-2 shows representative collections of models relevant to these areas.

Table 3-2. ASAC Models Used to Analyze AST Program Elements

AST program element	ASAC models
Advanced air transportation technology	ASAC Flight Segment Cost Model, ASAC Airport Capacity Model, ASAC Air Carrier Investment Model, ASAC System Safety Tolerance Analysis Model, National Airspace Research and Investment Model
Aging aircraft	ASAC Air Carrier Investment Model, ASAC Database
Civil tiltrotor	National Airspace Research and Investment Model, ASAC Database
Composites	Flight Optimization System Model, Aircraft Synthesis Model, ASAC Air Carrier Investment Model
Environmental assessment	ASAC Flight Segment Cost Model
Fly-by-light and power-by-wire	Flight Optimization System Model, Aircraft Synthesis Model, ASAC Air Carrier Investment Model
General aviation and commuter aviation	National Airspace Research and Investment Model, ASAC Database
Integrated wing	Flight Optimization System Model, Aircraft Synthesis Model, ASAC Air Carrier Investment Model
Propulsion	ASAC Flight Segment Cost Model, ASAC Air Carrier Investment Model, Flight Optimization System Model, Aircraft Synthesis Model
Terminal area productivity	ASAC Airport Capacity Model, ASAC Airport Delay Model, ASAC Air Carrier Investment Model, ASAC System Safety Tolerance Analysis Model, National Airspace Research and Investment Model

ASAC Model Integration Prototype

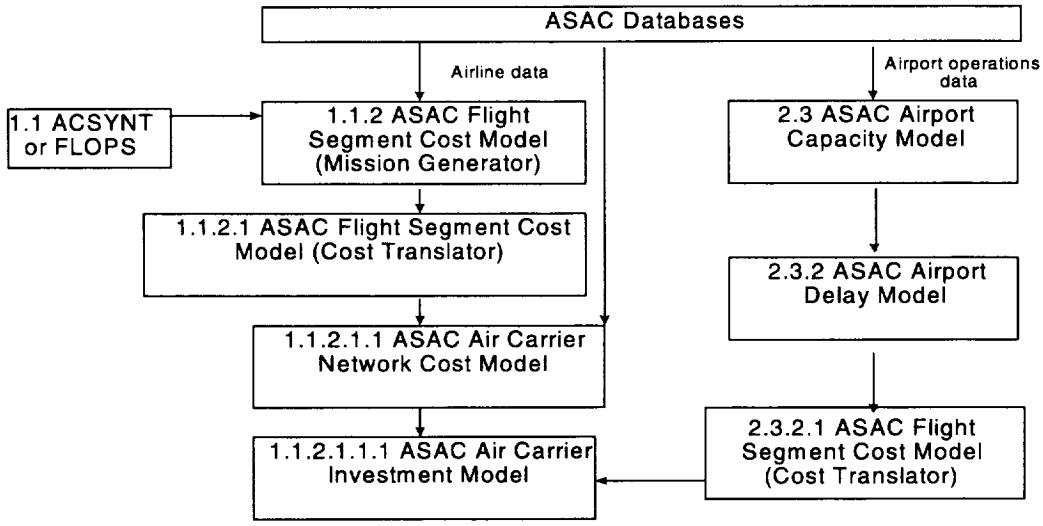
The ASAC Model Integration Prototype (First Generation ASAC), implemented in March 1997, was a subset of ASAC models. ASAC models can be linked together to form analyses. Two analyses are available; they are

- Aircraft Technology Analysis and
- Air Traffic Management Analysis.

Additional analyses will be added to the ASAC Model Integration Prototype (First Generation ASAC) as the models supporting them become available.

Figure 3-5 shows the models used in the ASAC Model Integration Prototype (First Generation ASAC). This collection of models enables analyses of improvements in aircraft technology (the left-most chain in Figure 3-5) or improvements in air traffic management (the right-most chain in Figure 3-5).

Figure 3-5. Models Used in the ASAC Model Integration Prototype



Chapter 4
Design Methodology

As discussed in the National Aeronautics and Space Administrator Contractor Report #201681, *ASAC Executive Assistant Architecture Description Summary,* the Domain-Specific Software Architecture (DSSA) is being used as a design methodology.

THE DOMAIN-SPECIFIC SOFTWARE ARCHITECTURE (DSSA) APPROACH

A domain engineering process is used to generate a DSSA. The goal of the process is to map user needs into system and software requirements that, based on a set of implementation constraints, eventually define a DSSA.

There are five stages in the DSSA domain engineering process. Each stage is further divided into steps or substages. The process is concurrent, recursive, and iterative. Therefore, completion requires several passes through each stage. The five stages in the domain engineering process are described in Table 4-1.

Table 4-1. DSSA Stages

Stage	Title	Description	ASAC EA phase
1	Define the scope of the domain	Definition of what can be accomplished with emphasis on user needs	Architecture
2	Define/refine domain-specific elements	Similar to requirements analysis with emphasis on the problem space	Architecture
3	Define/refine domain-specific design and implementation constraints	Similar to requirements analysis with emphasis on the solution space	Architecture
4	Develop domain models and architectures	Similar to high-level design with emphasis on defining module and model interfaces and semantics	Architecture and design
5	Produce and gather reusable work products	Implementation and collection of reusable artifacts such as code and documentation	Design and development

DSSA stages 1, 2, 3, and 4 (partial) were defined in the *ASAC Executive Assistant Architecture Description Summary*. DSSA stages 4 (remainder) and 5 (partial) are addressed in this document. The remainder of DSSA stage 5 will be addressed in ASAC development tasks, which will be follow-on efforts to this design task.

DSSA DESIGN TOOLS

Unified Modeling Language

Object oriented design (OOD) is a development approach based on the organization of entities that have structure and behavior. It promotes the construction of well-defined systems and facilitates reuse and ease of modification. The Object Modeling Technique (OMT), used to develop the *ASAC Executive Assistant Architecture Description Summary*, was one method used to cover the system development process from the conceptualization phase through implementation. The author of OMT has collaborated with the authors of other OOD methodologies, namely Booch and Jacobson, to create the Unified Modeling Language (UML). UML is the successor to the past object-oriented design notations and has been proposed as a standard to the Object Management Group (OMG). UML notation is used to document the design. The Rational Rose visual modeling tool is used to automate this process.

A brief description of UML diagrams is found in Table 4-2.

Table 4-2. UML Diagram Definitions

Diagram	Description
Use case	A snapshot of one aspect of a system. The sum of all use cases is the external picture of a system.
Sequence	An interaction diagram that models message passing behavior between objects.
Collaboration	An interaction diagram that models message passing behavior between objects.
Package	Shows a high-level picture of components (packaged classes) and the dependencies among them.
Class	A description of the classes in a system and the interrelationships among them.
State	Shows all possible states for an object and how the object's state changes as a result of events.
Deployment	Shows the physical relationships among software and hardware components in the delivered system.
Activity	Is a flow chart of tasks or methods on a class.
Data flow	A depiction of the relationships among functions, usually within the problem domain.

The first seven diagrams are used to represent the ASAC EA design in this document. The other two diagrams will be used in the future (they are not required at this point).

The methodology associated with the UML notation is called Objectory, and is still being developed. Like UML notation, Objectory brings together the best aspects of the OMT, Booch, and OOSE (Jacobson) methodologies.

Class-Responsibility-Collaboration (CRC) Card Technique

A technique called Class-Responsibility-Collaboration (CRC) Card is used to define the classes and class collaborations. CRC Card technique facilitates the process of discovering the real-world objects that make up a system and its public interfaces.

CRC cards are index cards that record

- suggested classes,
- their responsibilities,
  - what the classes know about themselves (knowledge responsibility)
  - what the classes do (behavior responsibility), and
- their relationship to other classes (collaboration).

CRC cards can optionally record

- class definitions and
- class attributes.

The front and optional back views of a CRC card are shown in Figures 4-1 and 4-2, respectively.

Figure 4-1. CRC Card—Front View

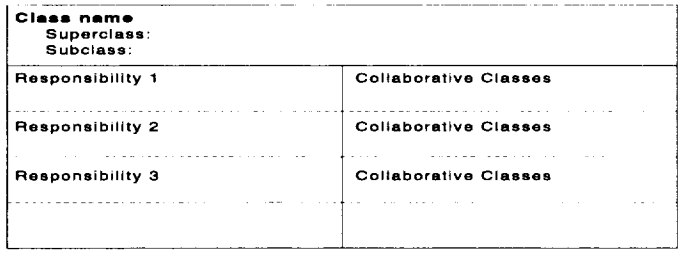


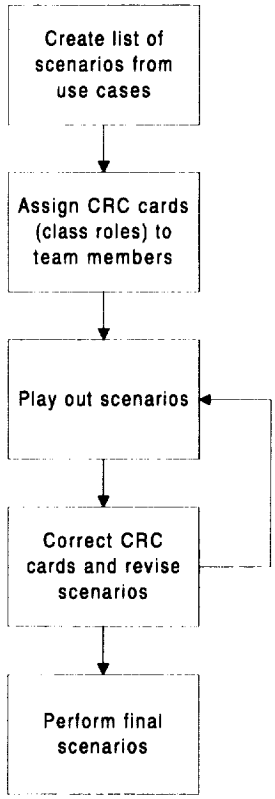
Figure 4-2. CRC Card—Back View (Optional)

The CRC cards are used to role-play system scenarios. A person represents a class and responds to a request from another class based upon what is written on his or her CRC card. The role-play enables one to

- validate classes,
- ensure the identification of what the class knows and what the class does, and
- ensure all class hierarchies are identified.

The CRC card process is depicted in Figure 4-3.

Figure 4-3. CRC Card Process



Design Patterns

Design patterns record experience in designing object-oriented software by naming, explaining, and evaluating important and recurring designs in object-oriented systems. An example of a design pattern is the Observer pattern, defined by Gamma, et al., as "a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically." The Observer, Flyweight, and Strategy design patterns were used in developing the ASAC EA design.

FURTHER READING

For more information on UML CRC cards and design patterns, see the following references:

[1] Fowler, Martin and Kendall Scott, "UML Distilled—Applying the Standard Object Modeling Language," Addison-Wesley, 1997.

[2] Rational Software Corporation UML Resource Center, "UML Document Set Version 1.1," September 1997, http://www.rational.com/uml/references/.

[3] Bellin, David and Susan Suchman Simone, "The CRC Card Book," Addison-Wesley, 1997.

[4] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides, "Design Patterns—Elements of Reusable Object-Oriented Software," Addison-Wesley, 1995.

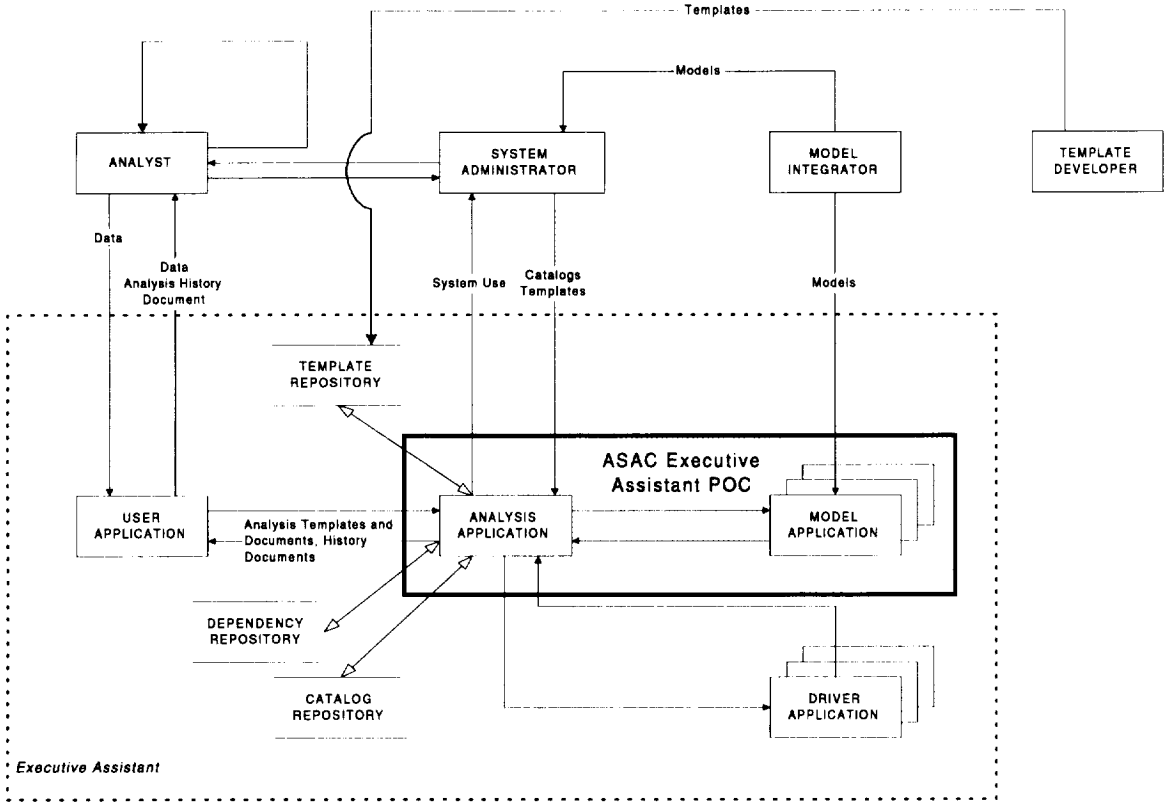
Chapter 5
ASAC EA Design

ASAC EA Proof of Concept

The DSSA approach was tailored to meet the needs of the ASAC design effort. This section discusses each of the applicable areas of the fourth and fifth DSSA stages.

This section is built upon the *ASAC Executive Assistant Architecture Description Summary*, which covered DSSA stages 1, 2, 3, and 4 (partial). It was developed to describe the general role of the ASAC system and give a general overview of the components of the system. A piece of the architecture, referred to as the ASAC Executive Assistant Proof of Concept (POC), is being developed to prove the concept of the system. Figure 5-1 shows the context diagram of the entire system. The POC portion of the entire system is shown by the highlighted box. This document will concentrate on the design of the POC.

Figure 5-1. Context Diagram



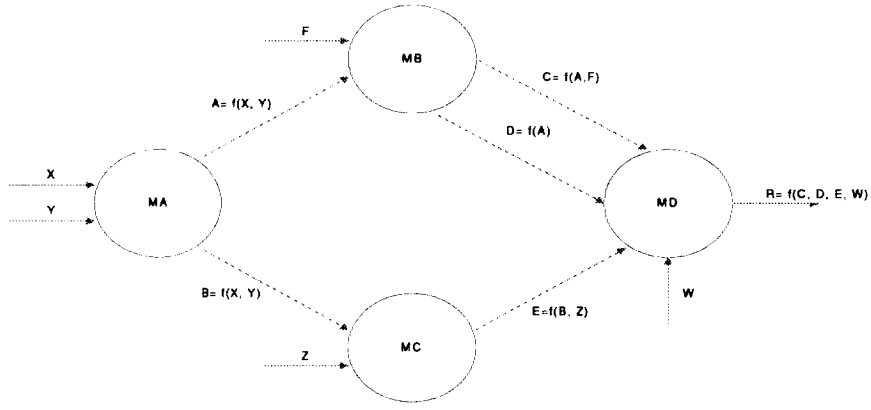
The ASAC Executive Assistant POC will perform an analysis that consists of multiple distributed models. The POC will perform the following:

- An analysis will be selected.
- The analysis will be given an input.
- The models and data relationships between the models and analysis will be constructed.
- The models will transform their input data into output data.
- When all transformations are finished, the analysis will be complete and the final output will be stored.

The analysis will communicate with the distributed models using Visigenic's implementation of the OMG Common Object Request Broker Architecture (CORBA) standard that makes the distributed nature of the models virtually transparent. The actual models will be wrapped by a standard interface defined using OMG IDL (Interface Definition Language) that allows the models to be distributed and provides clients a standard method for invoking all models. The analysis application and the model wrappers will be developed on the HP-UX platform using the C++ programming language. In addition, a mockup Java Graphical User Interface (GUI) client may be developed to prototype the end-user interface to the EA system.

Figure 5-2 shows the POC implementation. The POC analysis contains four models (MA, MB, MC, and MD), six data relationships (A, B, C, D, E, and R), and five user inputs (X, Y, Z, F, and W). This configuration was chosen because it exercises many of the characteristics for an analysis, i.e., single and multiple data relationships between models, single models feeding multiple models and multiple models feeding single models.

Figure 5-2. POC Implementation



REVIEW AND ITERATE DSSA STAGES 1 THROUGH 3

DSSA Substage 2-8: Define Assumptions

Additional assumptions defined during this design are as follows:

- Client will use an Internet Inter-ORB Protocol (IIOP) capable browser.
- When an object comes to life, it initializes itself and its parts, i.e., other objects do not explicitly have to tell an object to "initialize." If the actions of an object result in the creation of another object, then the "creator" object may pass some information along to the class instance to be created, which gives the created object some data for the initialization of this instance of the object.
- If a class A has the responsibility to "know" something, then class A has an encapsulated data store or structure that can contain instances of the data that it is to know. That data structure can be populated in one of three ways:
  - Some other class collaborates with Class A that results in an instance of Class A's data structure being created (or set).
  - When an instance of Class A is created, Class A is smart enough to retrieve the information that it needs in order to populate some/all of its encapsulated data structure.
  - A combination of i and ii.
- Models cannot commit suicide. They must be killed by some other class.
- Where we refer to an "Analysis," we refer to the combination of the Analysis and the Analysis Specification.
- When we refer to a "Model," we refer to the combination of a Model and a Model Specification. Furthermore, we do not care whether the "Model" is a legacy model (implies wrappers) or a new model.
- Model_Input_Data is a collection of data items. Each data item can either be "Set" or "Unset." A model cannot run until all of its input values are "Set." Note, NULL is a valid value for a "Set" data item.
- There are three instances of Model_Input_Data that a model maintains:
  - *Default_Input_Data*—When a model instance is created, it retrieves a "default" set of Model_Input_Data from the Model Catalog.

- *Initial_Input_Data*—Model_Input_Data that is taken from the Analysis Specification and the user.

- *Current_Input_Data*—The Current_Input_Data is the intersection of the Default_Input_Data and the Initial_Input_Data. Note, for each data item, the value of the Inital_Input_Data overrides that of the Default_Input_Data. Current_Input_Data also holds the most recent set of data that is used for determining when a model is ready for execution.

- Models and Analyses have at least the following states:

  - *Not_Ready_To_Run*—All Model_Input_Data have not been "set."

  - *Running*—Model is executing.

  - *Done*—Model has results available.

  - *Error*—Model has incurred some exception.

- Model_Output_Data—A collection of data items that are available from a model when a model is in the Done state.

- Model_Data_Names—Functional identifiers that categorize all data items in the system by type and allowable conversions.

- All Model_Input_Data items and Model_Output_Data items will have associated names. When a new model is added to the system, each of its Model_Input_Data items and Model_Output_Data items will be classified using a predefined name from an existing set of Model_Data_Names.

- At some point in the future, we will have a procedure/application for building a validated analysis specification. When we load an analysis, implicit in the specification is the execution order and model-to-model relationship. This holds for the predefined analysis that we will create for the POC.

- Living models can accept Model_Input_Data from at least the following sources:

  - User

  - Model Catalog

  - Analysis Specification

  - Another Model.

- A Model can never set another model's input data directly. Doing so would imply that models have knowledge of other models and their relationship.
- After a Model in the "Not_Ready_To_Run" state receives all of its required inputs, it executes on its own volition.
- After a Model has completed its execution, the model does not report its results. The model will notify someone else in the system that it is Done. And, that someone else will have to retrieve the results.
- Is the user ever presented with Model_Input_Data that they cannot change? No. The user may be presented with additional information specific to how the model is built or some information regarding model coefficients may be made available, but all Model_Input_Data will be modifiable by the user.
- The Analysis Specification is produced independently of the execution of the analysis, i.e., before an analysis can be executed, a completed Analysis Specification must be available. We can then think of the execution of the Analysis as a "batch" job.
- The system is event-driven.

DSSA Substage 2.9: Define Issues

Issues remaining from the ASAC Architecture are as follows:

- How does the EA system handle or detect nontermination of models?
- How is data passed among components? Pass data or data file name?
- Should multiple processes be spawned for the analysis application, or should there be separate invocations of the program?
- What are the space constraints on user systems (maximum size for the user application)?
- What is the target size of the analysis applications?

These issues will be addressed in future tasks.

DSSA Stage 4—Develop Domain Models

The goal for this phase of the domain-engineering is to develop domain models. This stage of the DSSA was based on OMT and has been revised to reflect the UML notation.

The following substages of DSSA stage 4 will be completed during the ASAC design effort:

- 4-1 Develop CRC cards
- 4-2 Develop the role-play script
- 4-3 Develop use case diagrams
- 4-4 Develop interaction diagrams
  - Sequence diagrams
  - Collaboration diagrams
- 4-5 Develop package diagrams
- 4-6 Develop class diagrams
- 4-7 Develop state diagrams
- 4-8 Develop deployment diagrams
- 4-9 Review and iterate

The beginning of the design phase requires classes to be defined. The classes extracted from the POC description are

- Subject,
- Observer,
- DataTransformer,
- Analysis,
- AnalysisSpecification,
- Model,
- ModelSpecification,
- DataRelationship,
- DataRelationshipSpecification,
- DataElement,
- DataElementSet,
- DataConverter, and
- DataStorage.

More detail on these classes will be provided throughout the DSSA Substages in this chapter.

DSSA Substage 4-1: Develop CRC Cards

CRC cards are a valuable object-oriented technique. They define superclasses and subclasses, the responsibilities of each class, and the collaboration among the classes. Tables 5-1 through 5-13 show the CRC cards for all of the classes.

Table 5-1. CRC Card for Subject Class

Subject	Design Pattern: Observer (293)—Gamma, et al.
superclasses:	
subclasses:	DataTransformer, DataElementSet, DataRelationship
Description: A superclass that defines the properties of an object being observed. A subject may have any number of dependent observers. All observers are notified when the subject undergoes a change in state.	
Know observers	
Notify observers when state changes	Observer

Table 5-2. CRC Card for Observer Class

Observer	Design Pattern: Observer (293)—Gamma, et al.
Superclasses:	
subclasses:	DataTransformer, DataRelationship
Description: Defines an updating interface for objects that should be notified of changes in a subject's state. In response to notification, observers query the subject to synchronize its state with the subject's state.	
Know subject	
Know state of subject	Subject
Register as an observer	Subject

Table 5-3. CRC Card for DataTransformer Class

DataTransformer	
superclasses:	Subject, Observer
subclasses:	Analysis, Model
Description: Abstraction for a class that transforms input data values into output data values.	
Know input data elements	DataElementSet
Know output data elements	DataElementSet
Notify observers when state changes	Observer
Know state	
Register as an observer	DataElementSet
Know state of input data elements	DataElementSet
Transform input values to output values	DataElementSet
Know Observers	

Table 5-4. CRC Card for Analysis Class

Analysis	
superclasses:	DataTransformer
subclasses:	
Description: Manages the creation and instantiation of models and their relationship.	
Create models and data relationships (Transform input values to output values)	(DataTransformer) AnalysisSpecification, Model, DataRelationship
Know input data elements	(DataTransformer) DataElementSet
Know output data elements	(DataTransformer) DataElementSet
Know status of models	Observer

Table 5-5. CRC Card for AnalysisSpecification Class

AnalysisSpecification	
superclasses:	DataStorage
subclasses:	
Description: Manages specification data for a particular analysis.	
Store analysis specification data	DataStorage
Retrieve analysis specification data	DataStorage
Know names of models	
Know execution points	
Know initial input data elements for Analysis	DataElementSet
Know initial output data elements for each model	DataElementSet
Know model relationships	
Know data relationships (between related models)	

Table 5-6. CRC Card for Model Class

Model	
superclasses:	DataTransformer
subclasses:	
Description: Represents the interface to a state of a model application.	
Know observers	
Notify observers when state changes	Observer
Know state	
Register as an observer	DataElementSet
Know state of input data elements	DataElementSet
Create relationship between self and another instance of a model	DataRelationship
Transform input values to output values	DataElementSet
Know input data elements	DataElementSet
Know output data elements	ModelSpecification

Table 5-7. CRC Card for ModelSpecification Class

ModelSpecification	
superclasses:	DataStorage
subclasses:	
Description: Manages specification data for a particular model.	
Know model name	
Know model description	
Know model location	
Know input file format	
Know output file format	
Know default data elements	DataElementSet
Know output data elements	DataElementSet

Table 5-8. CRC Card for DataRelationship Class

DataRelationship	
superclasses:	Observer, Subject
subclasses:	
Description: Observes a data source for changes in state. Gets data values from a model (when it is in particular state), converts the units if needed, and sets the values in a data target.	
Know input data elements	DataElementSet
Know output data elements	DataElementSet
Register as an observer	DataElementSet
Know state of input data elements	DataElementSet
Transform input data values to output data values	DataElementSet
Know relationship between input and output data elements	
Create a relationship between any DataTransformer subtype.	DataTransformer

Table 5-9. CRC Card for DataRelationshipSpecification Class

DataRelationshipSpecification	
superclasses:	DataStorage
subclasses:	
Description: Manages specification data for a particular model.	
Know data relationship name	
Know data relationship description	
Know data relationship location	
Know input file format	
Know output file format	

Table 5-10. CRC Card for DataElement Class

DataElement	
superclasses:	
subclasses:	
Description: A container for a piece of data.	
Know data name	
Know data type	
Know data unit	
Know data value	
Convert value to a different unit	DataConverter
Know state (set/unset)	

Table 5-11. CRC Card for DataElement Set Class

DataElementSet	
superclasses:	Subject, DataStorage
subclasses:	
Description: A collection of instances of DataElement.	
Know observers	
Notify observers when state changes	Observer
Know number of data elements	
Iterate through set of data elements	
Update value, state, name, type, or unit of the DataElements in a set	DataElement
Add data element to the set	
Know the state of the set	

Table 5-12. CRC Card for DataConverter Class

DataConverter	Design Patterns: Strategy (315), Flyweight (195)—Gamma, et al.
superclasses:	
subclasses:	
Description: Defines a common interface for converting a data value from one unit to another.	
Know source data unit	
Know source data value	
Know target data unit	
Know target data value	
Convert data value	

Table 5-13. CRC Card for DataStorage Class

DataStorage	
superclasses:	
subclasses:	AnalysisSpecification, ModelSpecification, DataRelationshipSpecification
Description: Manages storage and retrieval of data objects.	
Store data object	Security
Retrieve data object	Security
Know storage method	

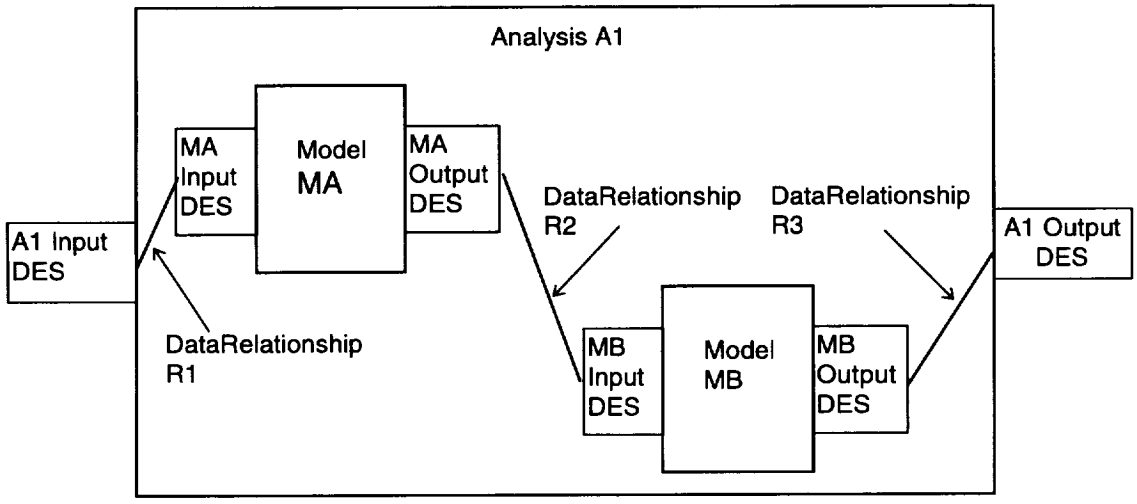
DSSA Substage 4-2: Develop the Role-Play Script

The role-play script used in the CRC Card technique enables the designers to talk through different scenarios that the program will execute. This further defines the role of each class. An example of a role-play script is shown below. This example is a reduced version of the POC to avoid repetition, however, it demonstrates all of the functionality of the POC design. The following is the scenario of the role-play:

- Build and execute a predefined **Analysis** named "**RP**" that consists of two known **Models**, named "**MA**" and "**MB.**"
- The output of "**MA**" is used as input to "**MB.**"
- There are initial values for the **Analysis** contained in a file "rp.val."
- The results of the **Analysis** are written to standard output.

Figure 5-3 illustrates the Analysis generated by the role-play script.

Figure 5-3. Role-Play Analysis



To achieve this scenario, the steps described in the next subsections are performed.

BUILDING AN ANALYSIS (A1)

1. **Main**: I request an instance of an analysis named "RP" from **Analysis**.

   Analysis: I provide an instance of **Analysis (A1)**.

2. **Analysis (A1)**: I request an instance of **AnalysisSpecification** for "RP" from **AnalysisSpecification**.

3. **AnalysisSpecification**: I create an instance of **AnalysisSpecification (AS1)** and initialize all values by requesting **AnalysisSpecification** data from **DataStorage**.

   DataStorage: I retrieve the **AnalysisSpecification** data and return it to **AnalysisSpecification (AS1)**.

   AnalysisSpecification: I return an instance of **AnalysisSpecification (AS1)** to **Analysis (A1)**.

4. **Analysis (A1)**: I request the input **DataElementSet** from my **Analysis-Specification (AS1)**.

   AnalysisSpecification (AS1): I return my input **DataElementSet**.

5. **Analysis (A1)**: I clone the returned **DataElementSet** and make the clone my input **DataElementSet (A1 Input)** and I request that it goes to the "Unset" state.

   DataElementSet (A1 Input): I go into the "Unset" state.

6. **Analysis (A1)**: I request a copy of the output **DataElementSet** from my **AnalysisSpecification**.

   AnalysisSpecification (AS1): I copy my output **DataElementSet** and return the copy.

7. **Analysis (A1)**: I take the returned **DataElementSet (A1 Output)** and make it my output **DataElementSet (A1 Output)** and I request that it goes to the "Unset" state.

   DataElementSet I go into the "Unset" state.

8. **Analysis (A1)**: I request to be attached as an observer to my input **Data ElementSet**.

   DataElementSet (A1 Input): I attach you as an observer.

9. **Main**: I request the input **DataElementSet** from **Analysis (A1).**

   Analysis (A1): I provide you with my input **DataElementSet**.

10. **Main**: I request the output **DataElementSet** from **Analysis (A1).**

    Analysis (A1): I provide you with my output **DataElementSet (A1 Input)**.

11. **Main**: I request that the **Analysis (A1)** input **DataElementSet (A1 Input)** read values from file "**rp.val**."

12. **DataElementSet (A1 Input)**: I request **DataStorage** to read file "**rp.val.**"

    DataStorage: I retrieve data and return it.

    DataElementSet (A1 Input): I populate my values and recalculate my state. I go to the "Set" state and notify my observers.

BUILDING A MODEL (MA)

1. **Analysis (A1)**: Observing that my input **DataElementSet (A1 Input)** is in the "Set" state, I request the name of a model from my **AnalysisSpecification**.

   AnalysisSpecification (AS1): There is a model named "MA."

2. **Analysis (A1)**: I request an instance of a model named "MA" from **Model**.

   Model: I provide you with an instance of **Model (MA)**.

3. **Model (MA)**: I request an instance of a **ModelSpecification** for "MA" from **ModelSpecification**.

4. **ModelSpecification**: I create an instance of **ModelSpecification (MSA)** and request **ModelSpecification** data from **DataStorage.**

   DataStorage: I retrieve **ModelSpecification** data and return it.

   ModelSpecification: I provide **Model (MA)** with an instance of **Model-Specification (MSA)**.

5. **Model (MA)**: I request my input **DataElementSet** from my **ModelSpecification (MSA)**.

   ModelSpecification (MSA): I return my input **DataElementSet**.

6. **Model (MA)**: I clone the returned **DataElementSet** and make it my input **DataElementSet (MA Input)**. I request that it go to the "Unset" state.

DataElementSet (MA Input): I go to the "Unset" state and notify my observers.

7. **Model (MA)**: I request my output **DataElementSet** from my **ModelSpecification (MSA)**.

   ModelSpecification (MSA): I return my output **DataElementSet** and return the copy.

8. **Model (MA)**: I clone the returned **DataElementSet** and make it my output **DataElementSet (MA Output)** and request that it go to the "Unset" state.

   DataElementSet (MA Output): I go to the "Unset" state.

9. **Model (MA)**: I request to be attached as an observer to my input **DataElementSet**.

   DataElementSet (MA Input): I attach you as an observer.

BUILDING ANOTHER MODEL (MB)

1. **Analysis (A1)**: Observing that my input **DataElementSet** is in the "Set" state, I request the name of a model from my **AnalysisSpecification**.

   AnalysisSpecification (AS1): There is a model named "MB."

2. **Analysis (A1)**: I request an instance of a model named "MB" from **Model**.

   Model: I provide you with an instance of **Model (MB).**

3. **Model (MB)**: I request an instance of a **ModelSpecification** for "MB" from **ModelSpecification.**

4. **ModelSpecification**: I create an instance of **ModelSpecification (MSB)** and request **ModelSpecification** data from **DataStorage.**

   DataStorage: I retrieve your **ModelSpecification** data and return it to you.

   ModelSpecification: I provide **Model (MB)** with an instance of **ModelSpecification (MSB).**

5. **Model (MB):** I request my input **DataElementSet** from my **ModelSpecification (MSB).**

   ModelSpecification (MSB): I return my input **DataElementSet.**

   Model (MB): I clone the returned **DataElementSet** and make it my input **DataElementSet (MB Input).** I request that it go to the "Unset" state.

DataElementSet (MB Input): I go to the "Unset" state and notify my observers.

6. **Model (MB)**: I request my output **DataElementSet** from my **ModelSpecification**.

   ModelSpecification (MSB): I return my output **DataElementSet**.

7. **Model (MB)**: I clone the returned **DataElementSet** and make it my output **DataElementSet (MB Output)** and request that it go to the "Unset" state.

   DataElementSet (MB Output): I go to the "Unset" state.

8. **Model (MB)**: I request to be attached as an observer to my input **DataElementSet**.

   DataElementSet (MB Input): I attach you as an observer.

BUILDING DATARELATIONSHIP (R1)

1. **Analysis (A1)**: I request the names of 2 data-related objects from my **AnalysisSpecification**.

   AnalysisSpecification (AS1): There is a data relationship between **Analysis (A1)** and **Model (MA)**.

2. **Analysis (A1)**: I request an instance of a **DataRelationship** named **R1** between **Analysis (A1)** and **Model (MA)**.

   DataRelationship: I provide an instance of a **DataRelationship (R1)**.

3. **DataRelationship (R1)**: I request an instance of a **DataRelationshipSpecification** for **Analysis (A1)** and **Model (MA)**.

4. **DataRelationshipSpecification**: I create an instance of **DataRelationshipSpecification (RS1)** and request default data from **DataStorage**.

   DataStorage: I retrieve **DataRelationshipSpecification** data and return it.

   DataRelationshipSpecification: I return the instance of **DataRelationshipSpecification (RS1)**.

5. **DataRelationship (R1)**: Knowing that **Analysis (A1)** is a parent and **Model (MA)** is a child, I ask **Analysis (A1)** for its input **DataElementSet**.

   Analysis (A1): I provide my input **DataElementSet (A1 Input)**.

DataRelationship (R1): I set my input **DataElementSet (R1 Input)** to the returned **DataElementSet**.

6. **DataRelationship (R1)**: Knowing that **Model (MA)** is a child of **Analysis (A1)**, I ask **Model (MA)** for its input **DataElementSet**.

   Model (MA): I provide my input **DataElementSet (MA Input)**.

   DataRelationship (R1): I set my output **DataElementSet (R1 Output)** to the returned **DataElementSet**.

7. **DataRelationship (R1)**: I request to be attached as an observer to my input **DataElementSet**.

   DataElementSet (R1 Input): I attach you as an observer.

8. **DataRelationship (R1)**: I request the state of my input **DataElementSet**.

   DataElementSet (R1 Input): I am in the "Unset" state.

BUILDING ANOTHER DATARELATIONSHIP (R2)

1. **Analysis (A1)**: I request the names of 2 data-related objects from my **AnalysisSpecification**.

   AnalysisSpecification (AS1): There is a data relationship between **Model (MA)** and **Model (MB)**.

2. **Analysis (A1)**: I request an instance of a **DataRelationship** named **R2** between **Model (MA)** and **Model (MB)**.

   DataRelationship: I provide an instance of **DataRelationship (R2)**.

3. **DataRelationship (R1)**: I request an instance of **DataRelationshipSpecification** for **Model (MA)** and **Model (MB)**.

4. **DataRelationshipSpecification**: I create an instance of **DataRelationshipSpecification (RS2)**, and request **DataRelationshipSpecification** data from **DataStorage**.

   DataStorage: I **retrieve** the **DataRelationshipSpecification** data and return it to you.

   DataRelationshipSpecification: I return the instance of **DataRelationshipSpecification (RS2)**.

5. **DataRelationship (R2)**: Knowing that **Model (MA)** and **Model (MB)** are children of the same parent, I ask **Model (MA)** for its output **DataElementSet**.

   Model (MA): I provide my output **DataElementSet (MA Output)**.

   DataRelationship (R2): I set my input **DataElementSet (R2 Input)** to the returned **DataElementSet**.

6. **DataRelationship (R2)**: Knowing that **Model (MA)** and **Model (MB)** are children of the same parent, I ask **Model (MB)** for its input **DataElementSet**.

   Model (MB): I provide my input **DataElementSet (MB Input)**.

   DataRelationship (R2): I set my output **DataElementSet (R2 Output)** to the returned **DataElementSet**.

7. **DataRelationship (R2)**: I request to be attached as an observer to my input **DataElementSet**.

   DataElementSet (R2 Input): I attach you as an observer.

8. **DataRelationship (R2)**: I request the state of my input **DataElementSet**.

   DataElementSet (R2 Input): I am in the "Unset" state.

BUILDING ANOTHER DATARELATIONSHIP (R3)

1. **Analysis (A1)**: I request the names of 2 data-related objects from my **AnalysisSpecification**.

   AnalysisSpecification (AS1): There is a data relationship between **Model (MB)** and **Analysis (A1)**.

2. **Analysis (A1)**: I request an instance of a **DataRelationship** named **R3** between **Model (MB)** and **Analysis (A1)**.

   DataRelationship: I provide an instance of **DataRelationship (R3)**.

3. **DataRelationship (R3)**: I request an instance of **DataRelationshipSpecification** for **Model (MB)** and **Analysis (A1).**

4. **DataRelationshipSpecification**: I create an instance of **DataRelationshipSpecification (RS3)** and request **DataRelationshipSpecification** data from **DataStorage**.

DataStorage: I retrieve the **DataRelationshipSpecification** data and return it to you.

DataRelationshipSpecification: I return the instance of **DataRelationshipSpecification (RS3)**.

5. **DataRelationship (R3)**: Knowing that **Model (MB)** is a child of the parent **Analysis (A1)**, I ask **Model (MB)** for its output **DataElementSet**.

   Model (MB): I provide my output **DataElementSet (MA Output)**.

   DataRelationship (R3): I set my input **DataElementSet (R3 Input)** to the returned **DataElementSet**.

6. **DataRelationship (R3)**: Knowing that **Analysis (A1)** is a parent and **Model (MB)** is a child, I ask **Analysis (A1)** for its output **DataElementSet**.

   Analysis (A1): I provide my output **DataElementSet (A1 Output)**.

   DataRelationship (R3): I set my output **DataElementSet (R3 Output)** to the returned **DataElementSet**.

7. **DataRelationship (R3)**: I request to be attached as an observer to my input **DataElementSet**.

   DataElementSet (R3 Input): I attach you as an observer.

8. **DataRelationship (R3)**: I request the state of my input **DataElementSet**.

   DataElementSet (R3 Input): I am in the "Unset" state.

   Analysis (A1): I request the names of 2 more data-related objects from my **AnalysisSpecification**.

   AnalysisSpecification (AS1): There are no more data-related objects.

RUNNING THE ANALYSIS (A1)

1. **DataRelationship (R1)**: Observing that my input **DataElementSet** is in the "Set" state, I transform my input **DataElementSet** values and apply them to my output **DataElementSet**.

2. **DataElementSet (R1 Output)**: I update my values and reevaluate my state. I change to the "Set" state and notify my observers.

3. **Model (MA)**: Observing that my input **DataElementSet** is in the "Set" state, I change to "Running" state, I transform my input **DataElementSet**

values and apply them to my output **DataElementSet**, and change to the "Done" state.

4. **DataElementSet (MA Output)**: I update my values and reevaluate my state. I change to the "Set" state and notify my observers.

5. **DataRelationship (R2)**: Observing that my input **DataElementSet** is in the "Set" state, I transform my input **DataElementSet** values and apply them to my output **DataElementSet**.

6. **DataElementSet (R2 Output)**: I update my values and reevaluate my state. I change to the "Set" state and notify my observers.

7. **Model (MB)**: Observing that my input **DataElementSet** is in the "Set" state, I change to "Running" state, I transform my input **DataElementSet** values and apply them to my output **DataElementSet**, and change to the "Done" state.

8. **DataElementSet (MB Output)**: I update my values and reevaluate my state. I change to the "Set" state and notify my observers.

9. **DataRelationship (R3)**: Observing that my input **DataElementSet** is in the "Set" state, I transform my input **DataElementSet** values and apply them to my output **DataElementSet**.

   DataElementSet (R3 Output): I update my values and reevaluate my state. I change to the "Set" state and notify my observers.

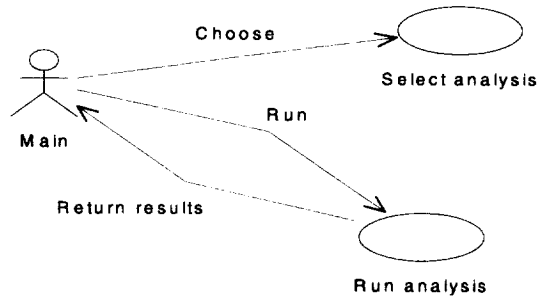
   Main: While polling the state of **Analysis (A1)** output **DataElementSet (A1 Output)** I see that the output **DataElementSet (A1 Output)** is now "Set." I request that its output be written.

   DataElementSet (R3 Output): I write my values.

DSSA Substage 4-3: Develop Use Case Diagrams

Use Case diagrams are used to show a typical interaction between a user and the system. The Use Case diagram for the POC is shown in Figure 5-4. It illustrates that a user will be able to choose an analysis, run an analysis, and obtain the results from the analysis.

Figure 5-4. POC Use Case Diagram



DSSA Substage 4-4: Develop Interaction Diagrams

Interaction diagrams are diagrams that describe how groups of objects collaborate. These diagrams usually capture the behavior of a single Use Case. The two types of Interaction diagrams are sequential diagrams and collaboration diagrams. Sequential diagrams and Collaboration diagrams give the same temporal information, but are shown in two different ways. Objects in a sequence diagram are shown as a box with a dashed line below it that represents the objects lifeline.

Each message is represented by an arrow between two lifelines. Objects in a collaboration diagram are shown as icons and the message is represented by arrows between two icons.

BUILDING AN ANALYSIS

Refer to the role-play script **Building an Analysis (A1)** when viewing Figure 5-5 and 5-6.

Figure 5-5. Building An Analysis Sequence Diagram

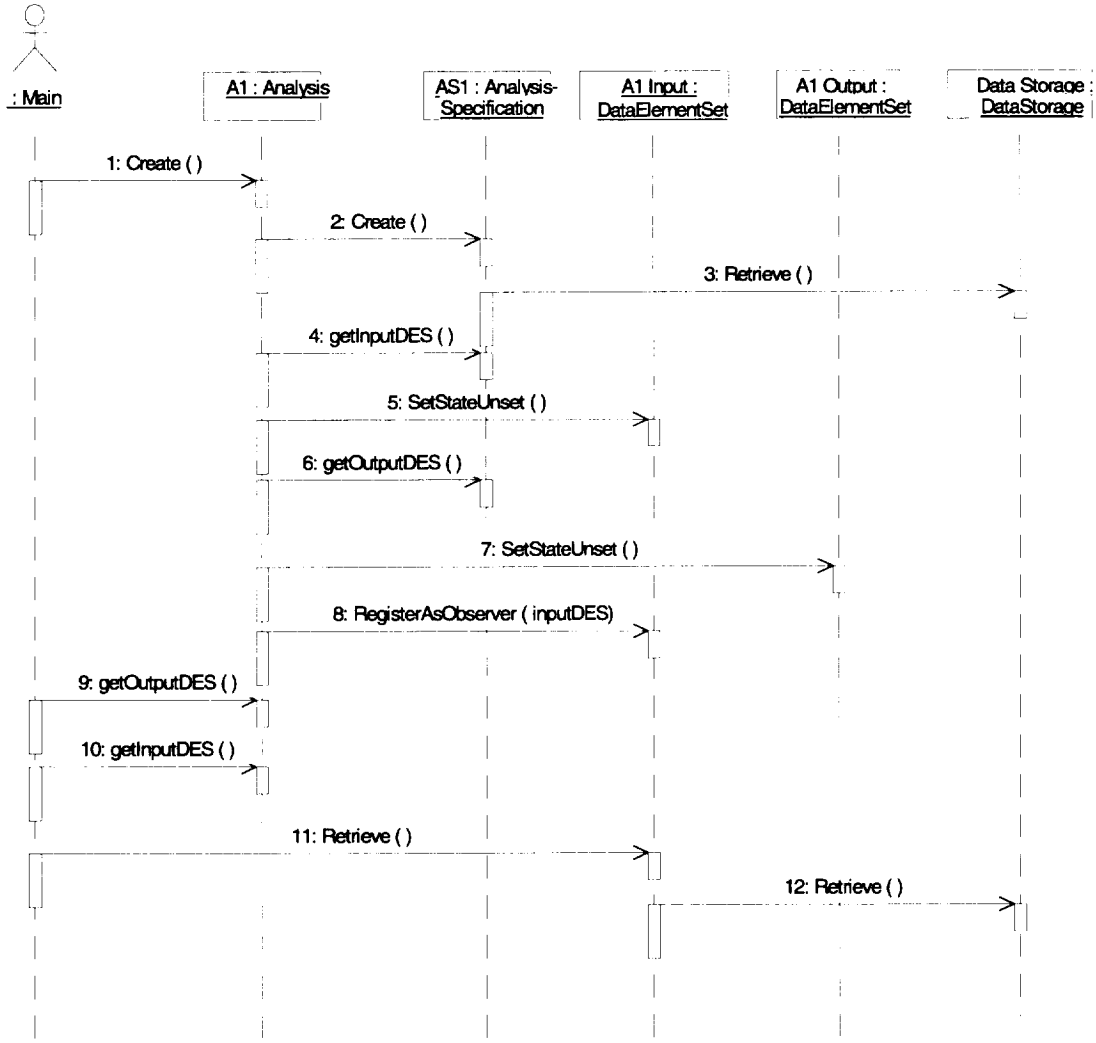
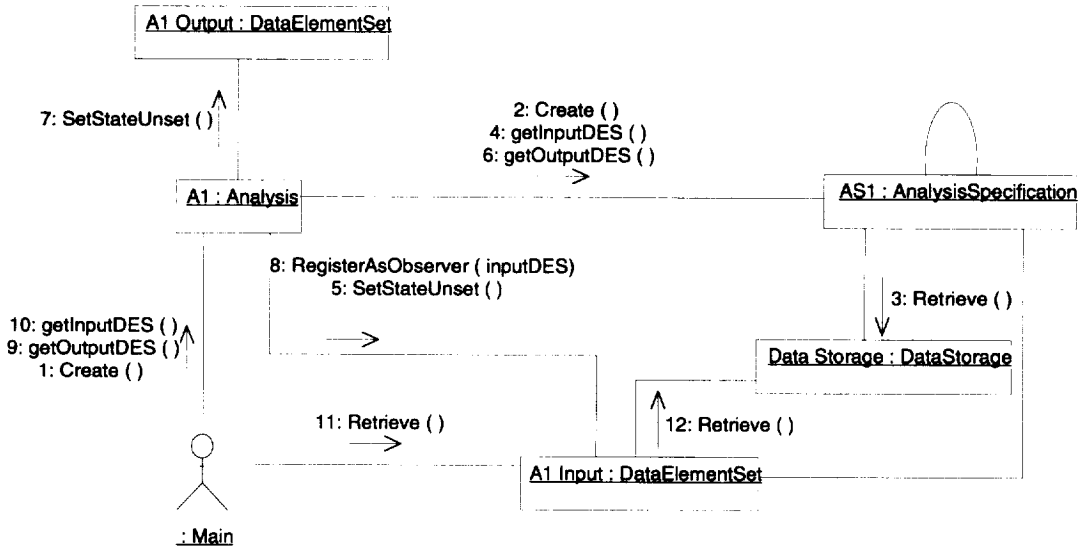


Figure 5-6. Building An Analysis Collaboration Diagram



BUILDING A MODEL

Refer to the role-play script **Building a Model (MA)** when viewing Figures 5-7 and 5-8.

Figure 5-7. Building a Model Sequence Diagram

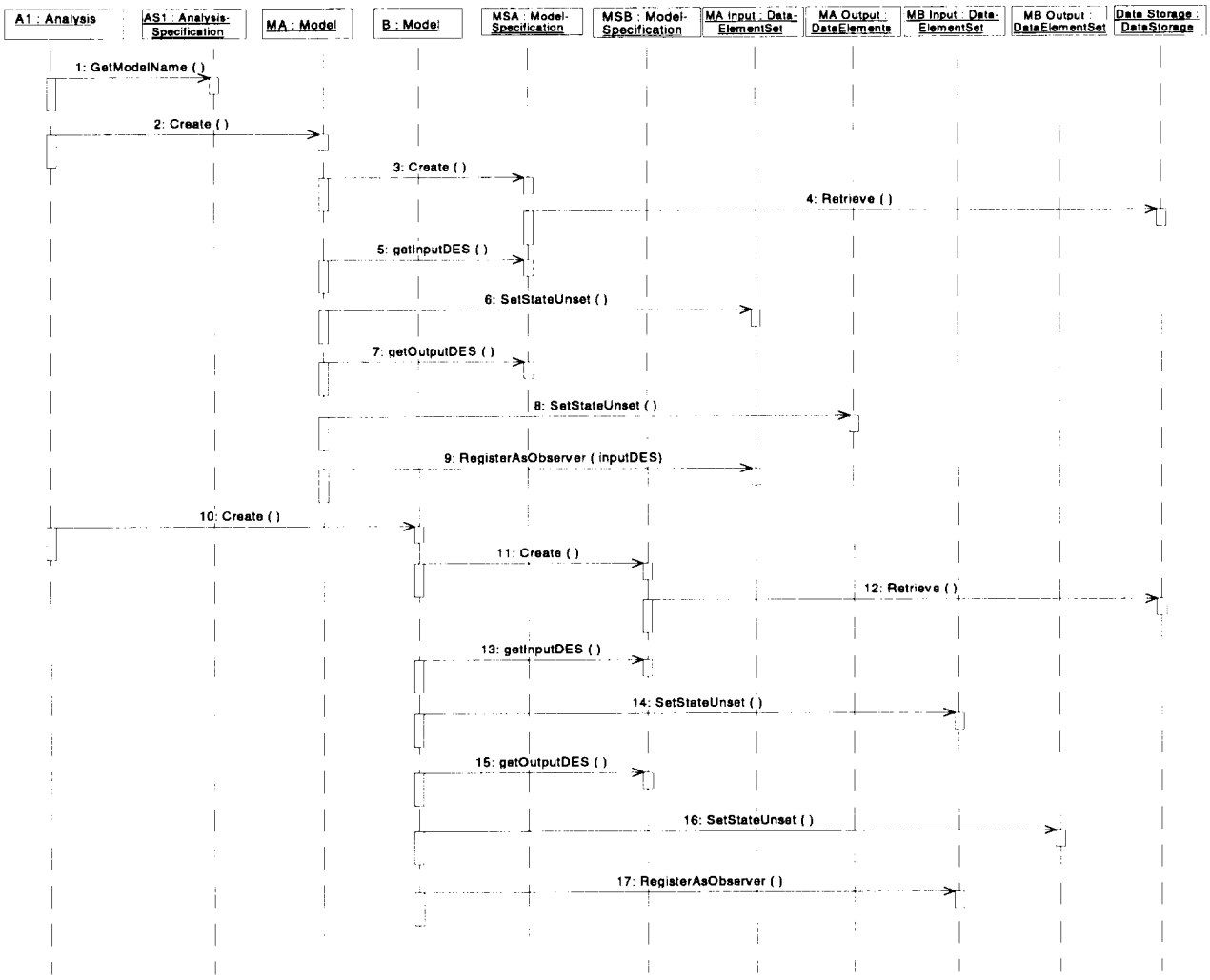
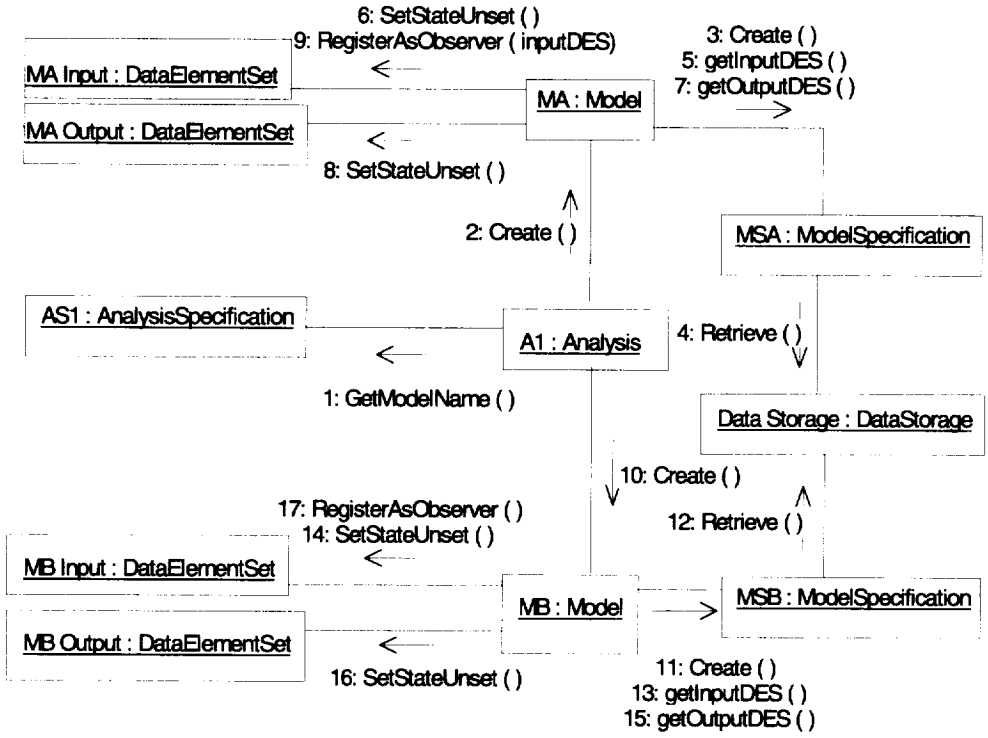


Figure 5-8. Building a Model Collaboration Diagram



BUILDING A DATARELATIONSHIP BETWEEN AN ANALYSIS AND A MODEL

Refer to the role-play script **Building a DataRelationship (R1)** when viewing Figures 5-9 and 5-10.

Figure 5-9. Building a DataRelationship Between an Analysis and a Model Sequence Diagram

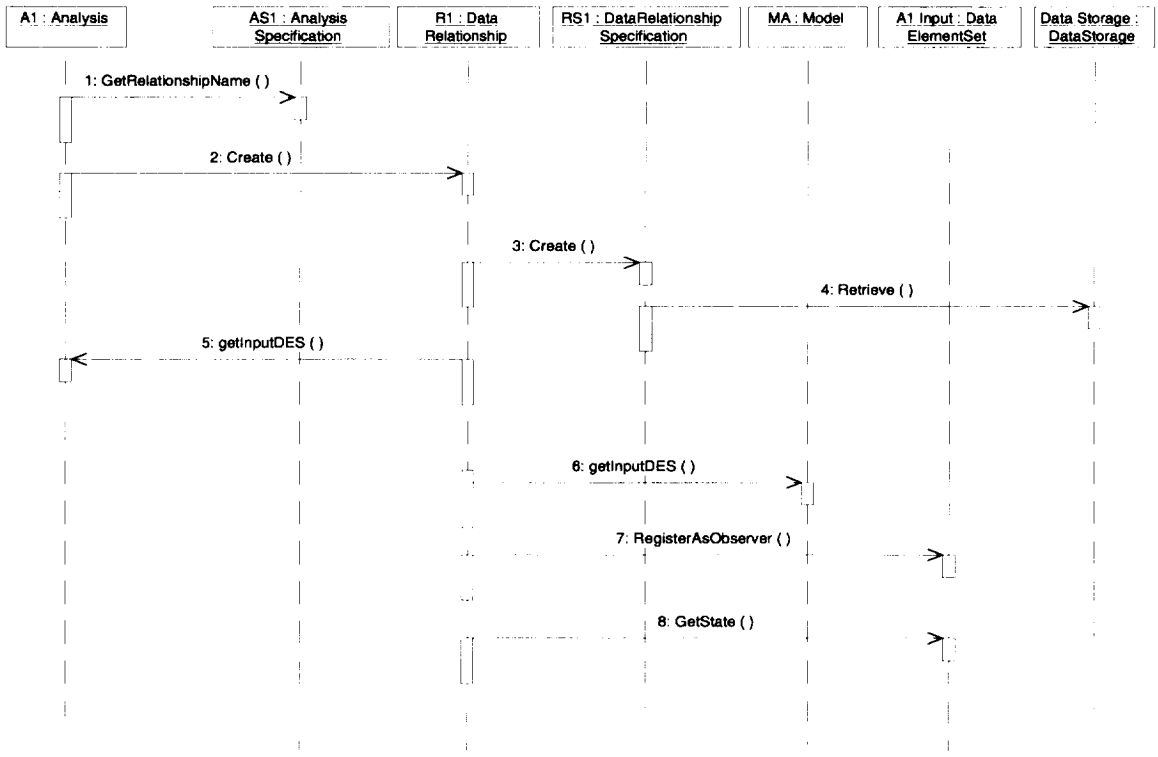
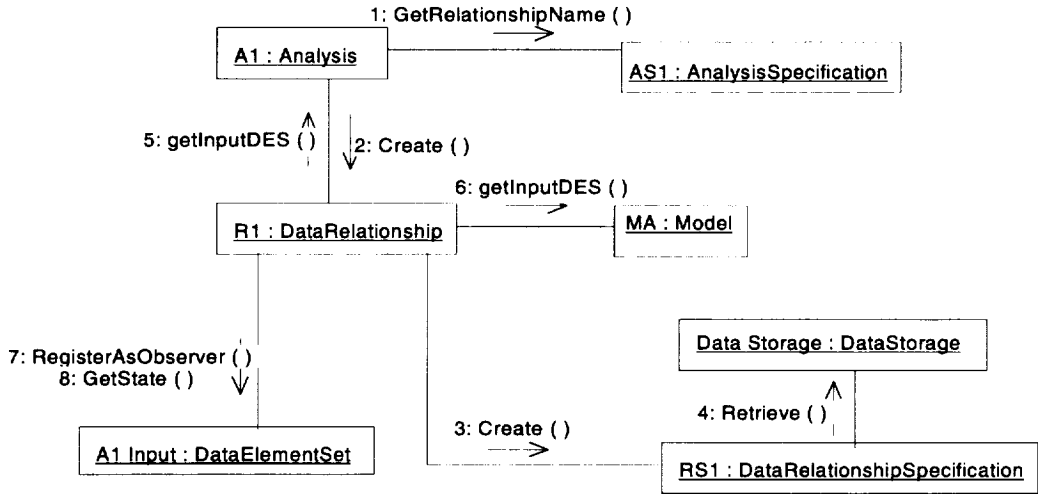


Figure 5-10. Building a DataRelationship Between an Analysis and a Model Collaboration Diagram



BUILDING A DATARELATIONSHIP BETWEEN TWO MODELS

Refer to the role-play script **Building A DataRelationship (R2)** when viewing Figures 5-11 and 5-12.

Figure 5-11. Building a DataRelationship Between a Model and a Model Sequence Diagram

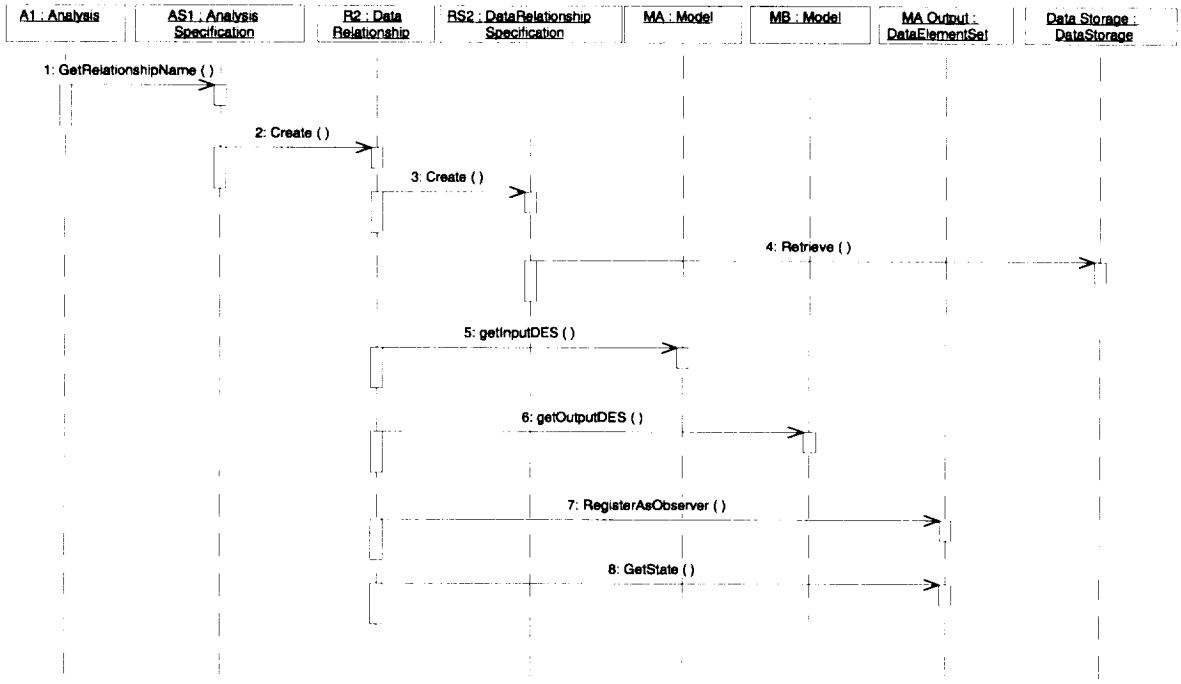
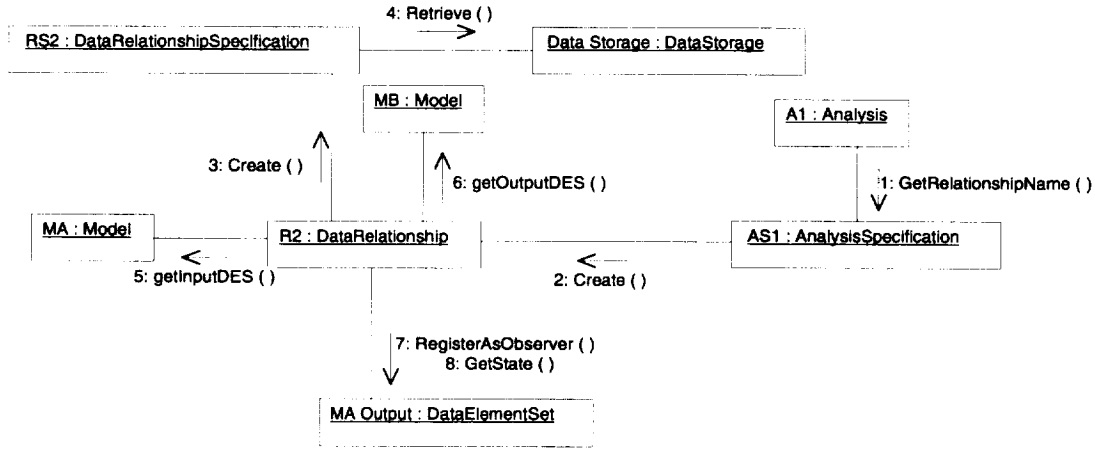


Figure 5-12. Building a DataRelationship Between a Model and a Model Collaboration Diagram



BUILDING A DATARELATIONSHIP BETWEEN A MODEL AND AN ANALYSIS

Refer to the role-play script **Building A DataRelationship (R3)** when viewing Figures 5-13 and 5-14.

Figure 5-13. Building a DataRelationship Between a Model and an Analysis Sequence Diagram

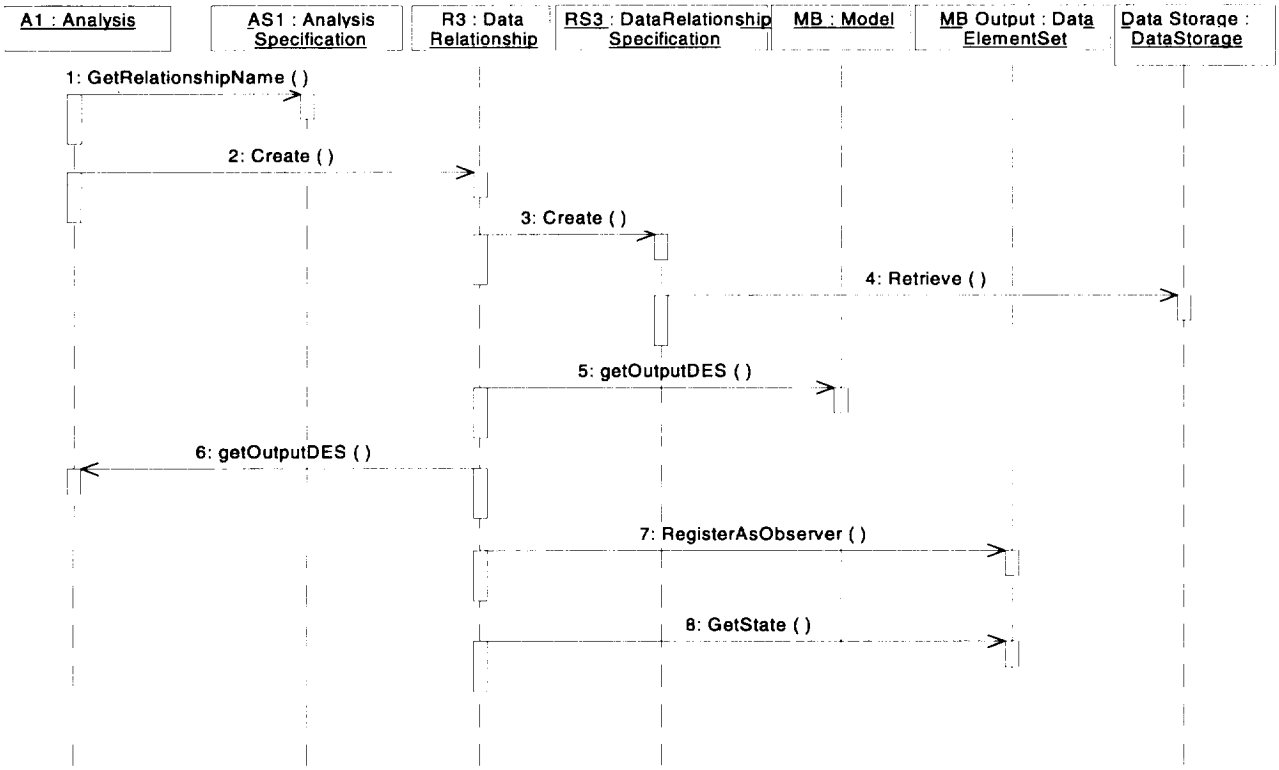
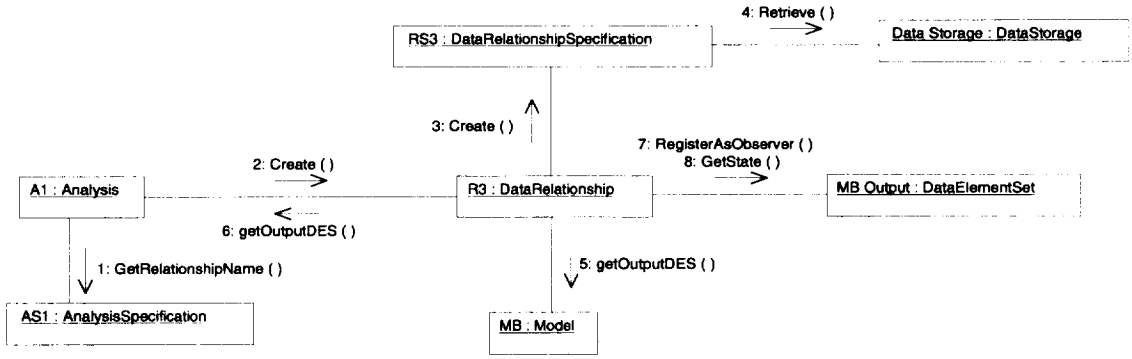


Figure 5-14. Building a DataRelationship Between a Model and an Analysis Collaboration Diagram



Refer to the role-play script **Running the Analysis (A1)** when viewing Figures 5-15 and 5-16.

Figure 5-15. Running the Analysis Sequence Diagram

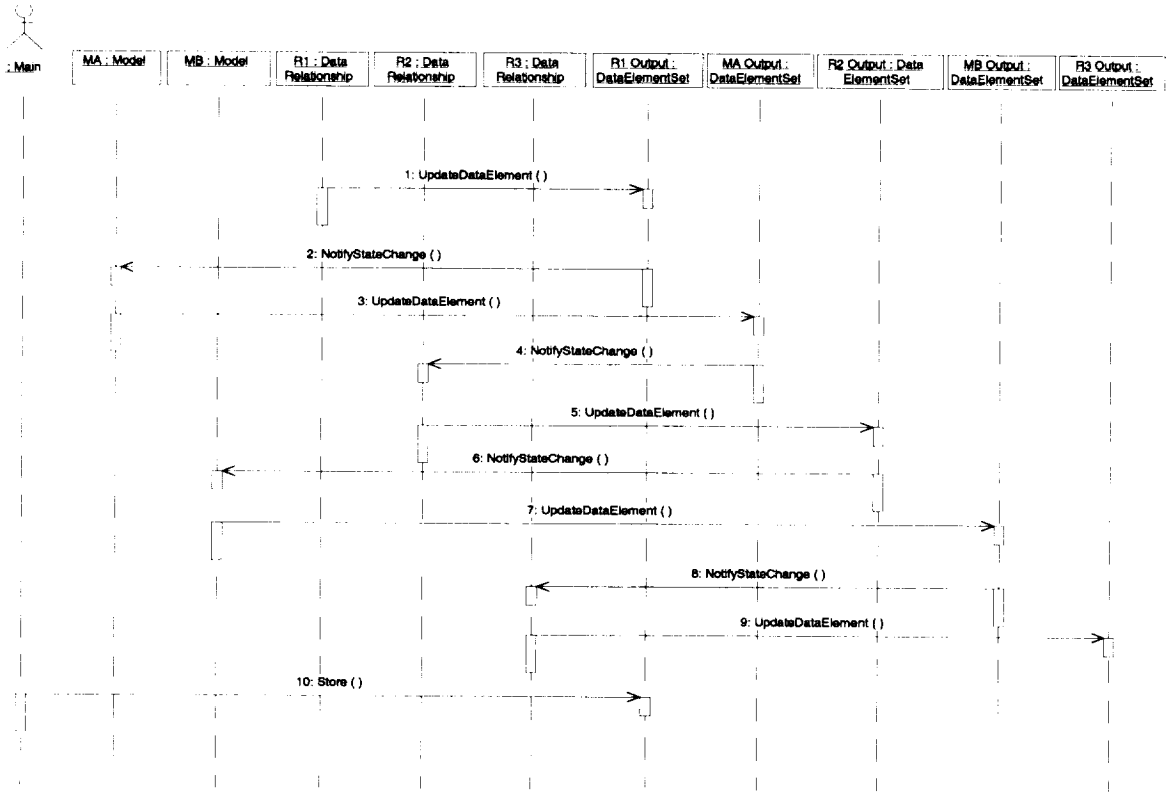
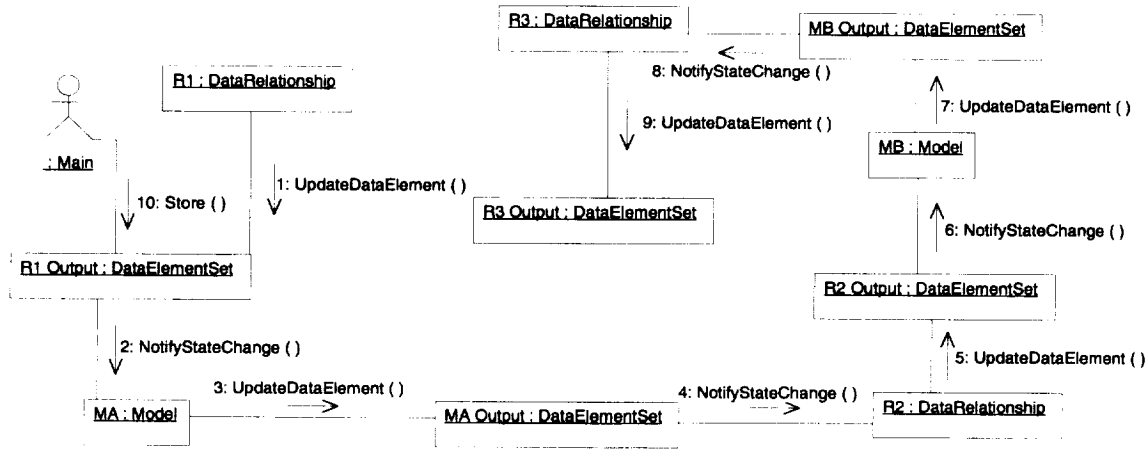


Figure 5-16. Running The Analysis Collaboration Diagram



DSSA Substage 4-5: Develop Package Diagrams

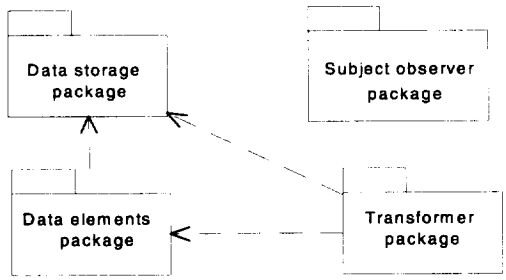
Package diagrams are used for readability purposes only. When a design becomes large, it is convenient to separate groups of classes into separate packages. The POC design has been divided into four class packages:

- Subject Observer package
- Transformer package
- Data Element package
- Data Storage package.

Figure 5-17 shows the POC package diagram. The dependencies among the classes are denoted by the dashed lines. These dependencies are the following:

- The Data Elements package depends on the Data Storage package to store and retrieve input and output data.
- The Transformer package depends on the Data Storage package to store and retrieve specification data.
- The Transformer package depends on the Data Elements package to supply DataElements for its transformers.

Figure 5-17. Package Diagram



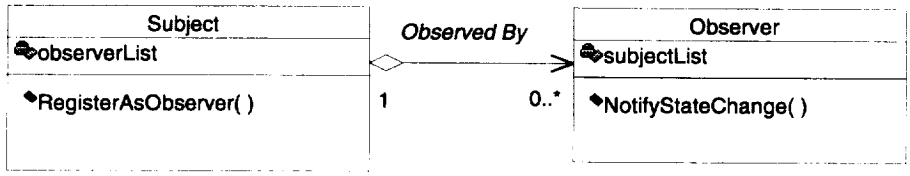
DSSA Substage 4-6: Develop Class Diagrams

Class diagrams are used to illustrate class models and their relationships with other classes. The class diagrams will be shown in accordance with their package.

SUBJECT OBSERVER PACKAGE

The Subject Observer package contains the Subject class and Observer class. The Subject Observer class diagram indicates that one subject will be observed by zero or more observers. The class diagram is shown in Figure 5-18.

Figure 5-18. Subject Observer Class Diagram



Subject

The subject is a superclass that defines the properties of an object being observed. A subject may have any number of dependent observers. All observers are notified when the subject undergoes a change in state. A list of properties and methods for this class can be found in Table 5-14.

Table 5-14. Properties and Methods for Subject Class

Private Properties	
observerList:	The observerList attribute is a list of Observers.
Public Methods	
RegisterAsObserver ()	The RegisterAsObserver operation is used to allow subjects to register as observers.

Observer

The Observer defines an updating interface for objects that should be notified of changes in a subject's state. In response to notification, observers query the subject to synchronize its state with the subject's state. A list of properties and methods for this class can be found in Table 5-15.

Table 5-15. Properties and Methods for Observer Class

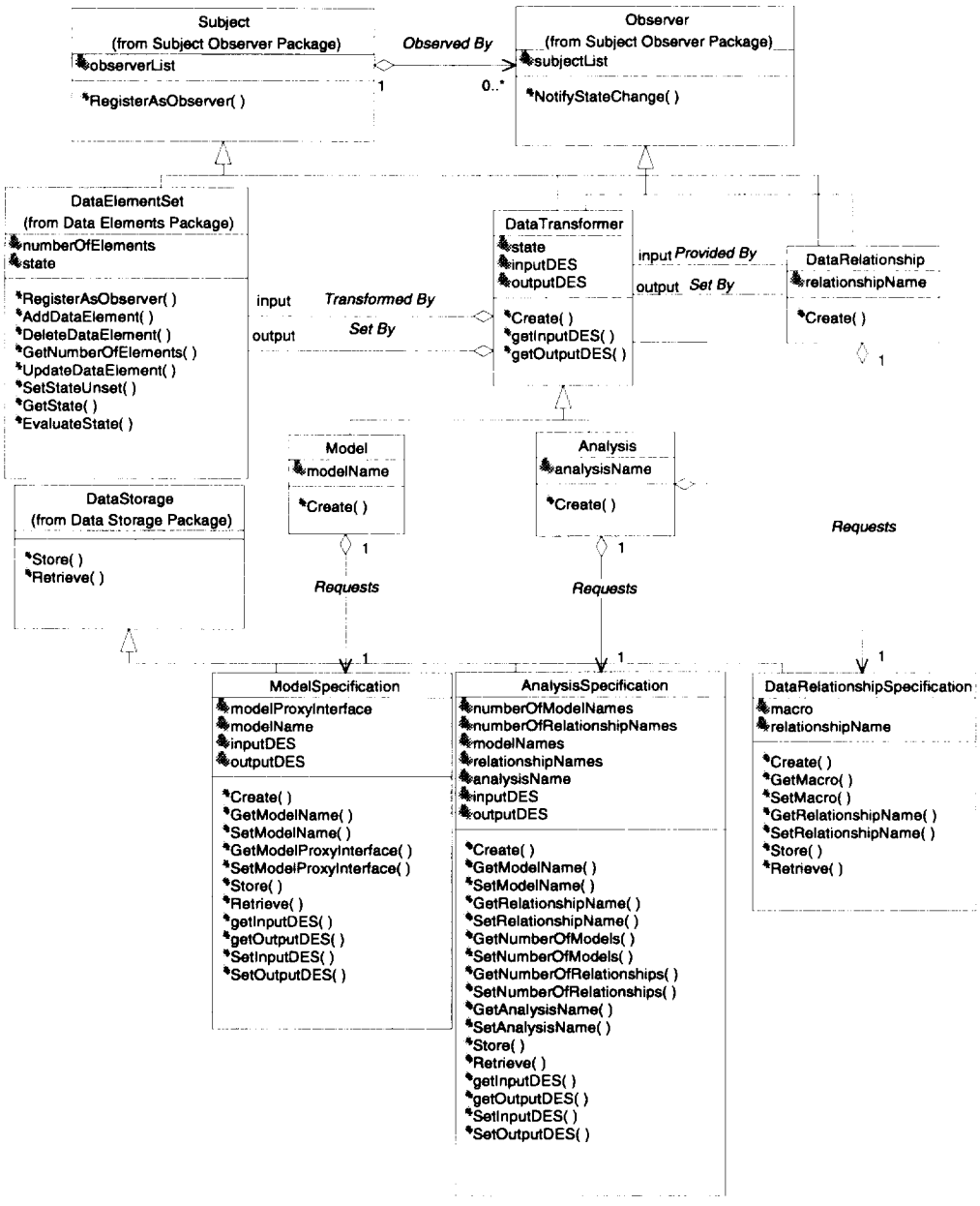
Private Properties	
subjectList	The subjectList attribute is a list of subjects.
Public Methods	
NotifyStateChange ()	The NotifyStateChange operation is used to notify Observers of a state change in a Subject.

TRANSFORMER PACKAGE

The Transformer package contains the associations between the data transformers, i.e., the Analysis, Models and DataRelationships, and their specifications. The Transformer class diagram is shown in Figure 5-19. The Subject, Observer Data Storage, and DataElementSet classes are shown in this package to illustrate their relationship with the transformers and their specifications. They are not a part of

the Transformer Package. All DataTransformers inherit from the Subject and Observer classes. The Analysis is composed of many other Analysis, Models, and DataRelationships. Each Analysis, Model, and DataRelationship requests information containe in their AnalysisSpecification, ModelSpecification and DataRelationshipSpecification respectively. All specifications use DataStorage to store and retrieve their contents. The input DataElementSet is transformed by the DataTransformers and the output DataElementSets are set by the Data Transformers.

Figure 5-19. Transformer Class Diagram



DataTransformer

The DataTransformer is an abstraction for a class that transforms input data values into output data values. A list of properties and methods for this class can be found in Table 5-16. The DataTransformer inherits from the Subject and Observer classes, so it also contains the properties and methods shown in Tables 5-14 and 5-15.

Table 5-16. Properties and Methods For DataTransformer Class

Private Properties	
state	The state attribute defines the state of the DataTransformer.
InputDES	The inputDES attribute contains the input DataElementSet of the DataTransformer.
OutputDES	The outputDES attribute contains the output DataElementSet of the DataTransformer.
Public Methods	
Create ()	The Create operation is used to create a DataTransformer
getInputDES ()	The getInputDES operation is used to get the input DataElementSet.
getOutputDES ()	The getOutputDES operation is used to get the output DataElementSet.

Analysis

The Analysis class manages the creation and instantiation of models and their data relationships. A list of properties and methods for this class can be found in Table 5-17. The Analysis class inherits from the DataTransformer class, so it also contains the properties and methods shown in Tables 5-14, 5-15, and 5-16.

Table 5-17. Properties and Methods for Analysis Class

Private Properties	
analysisName	The analysisName attribute contains the name of the analysis.
Public Methods	
Create ()	The Create operation is used to create the analysis.

AnalysisSpecification

The AnalysisSpecification manages specification data for a particular analysis. A list of properties and methods for this class can be found in Table 5-18.

Table 5-18. Properties and Methods for AnalysisSpecification Class

Private Properties	
numberOfModelNames	The numberOfModelNames attribute contains the number of Models in the Analysis.
numberOfRelationshipNames	The numberOfRelationshipNames attribute contains the number of DataRelationships in an Analysis.
modelNames	The modelNames attribute contains the names of Models in the Analysis.
relationshipNames	The relationshipNames attribute contains the names of DataRelationships in the Analysis.
analysisName	The analysisName attribute contains the analysis name.
inputDES	The inputDES attribute contains the input DataElementSet for the Analysis.
outputDES	The outputDES attribute contains the output DataElementSet for the Analysis.
Public Methods	
Create ()	The Create operation is used to create the AnalysisSpecification.
GetModelName ()	The outputDES operation is used to get Model names from the Analysis.
SetModelName ()	The SetModelName operation is used to set the Model names in an Analysis.
GetRelationshipName ()	The GetRelationshipName operation is used to GetDataRelationship names from the Analysis.
SetRelationshipName ()	The SetRelationshipName operation is used to set the DataRelationship names in an Analysis.
GetNumberOfModels ()	The GetNumberOfModels operation is used to get the number of Models in an Analysis.
SetNumberOfModels ()	The SetNumberOfModels operation is used to set the number Models in an Analysis.
GetNumberOfRelationships ()	The GetNumberOfRelationships operation is used to get the number of DataRelationships in an Analysis.
SetNumberOfRelationships ()	The SetNumberOfRelationships operation is used to set the number of DataRelationships in an Analysis.
GetAnalysisName ()	The GetAnalysisName operation is used to get the Analysis name.
SetAnalysisName ()	The SetAnalysisName operation is used to set the Analysis name.
Store ()	The Store operation is used to store the contents of the AnalysisSpecification.
Retrieve ()	The Retrieve operation is used to retrieve the contents of the AnalysisSpecification.
getInputDES ()	The getInputDES operation is used to get the Input DataElementSet for the Analysis.
getOutputDES ()	The getOutputDES operation is used to get the output DataElementSet for the Analysis.
SetInputDES ()	The SetInputDES operation is used to set the input Data ElementSet for the Analysis.
SetOutputDES ()	The SetOutputDES operation is used to set the output Data ElementSet for the Analysis.

Model

The Model class represents the interface to a state of a model application. A list of properties and methods for this class can be found in Table 5-19. The Model class inherits from the DataTransformer class, so it also contains the properties and methods shown in Tables 5-14, 5-15, and 5-16.

Table 5-19. Properties and Methods for Model Class

Private Properties	
modelName	The modelName attribute contains the Model name.
Public Methods	
Create ()	The Create operation is used for creating the Model.

ModelSpecification

The ModelSpecification manages specification data for a particular model. A list of properties and methods for this class can be found in Table 5-20.

Table 5-20. Properties and Methods for ModelSpecification Class

Private Properties	
modelProxyInterface	The modelProxyInterface attribute contains the proxy interface.
modelName	The modelName attribute contains the model name.
inputDES	The inputDES attribute contains the Models input DataElementSet.
outputDES	The outputDES attribute contains the Models output DataElementSet.
Public Methods	
Create ()	The Create operation is used to create the ModelSpecification.
GetModelName ()	The GetModelName operation is used to get the Model names.
SetModelName ()	The SetModelName operation is used to set the Model names.
GetModelProxyInterface ()	The GetModelProxyInterface operation is used to get the Model proxy.
SetModelProxyInterface ()	The SetModelProxyInterface operation is used to set the Model proxy.
Store ()	The Store operation is used to store the contents of the ModelSpecification.
Retrieve ()	The Retrieve operation is used to retrieve the contents of the ModelSpecification.
GetInputDES ()	The GetInputDES operation is used to get the input DataElementSet for the Model.
GetOutputDES ()	The GetOutputDES operation is used to get the output DataElementSet for the Model.
SetInputDES ()	The SetInputDES operation is used to set the input DataElementSet for the Model.
SetOutputDES ()	The SetOutputDES operation is used to set the output DataElementSet for the Model.

DataRelationship

The DataRelationship class observes a data source for changes in state, gets data values from a model (when it is in particular state), converts the units if needed, and sets the values in a data target. A list of properties and methods for this class can be found in Table 5-21. DataRelationship inherits from the Subject and Observer classes, so it also contains the properties and methods shown in Tables 5-14 and 5-15.

Table 5-21. Properties and Methods for DataRelationship Class

Private Properties	
relationshipName	The relationshipName attribute contains the DataRelationship name.
Public Methods	
Create ()	The Create operation is used to create the DataRelationshipSpecification.

DataRelationshipSpecification

The DataRelationshipSpecification manages specification data for a particular data relationship. A list of properties and methods for this class can be found in Table 5-22.

Table 5-22. Properties and Methods for DataRelationshipSpecification Class

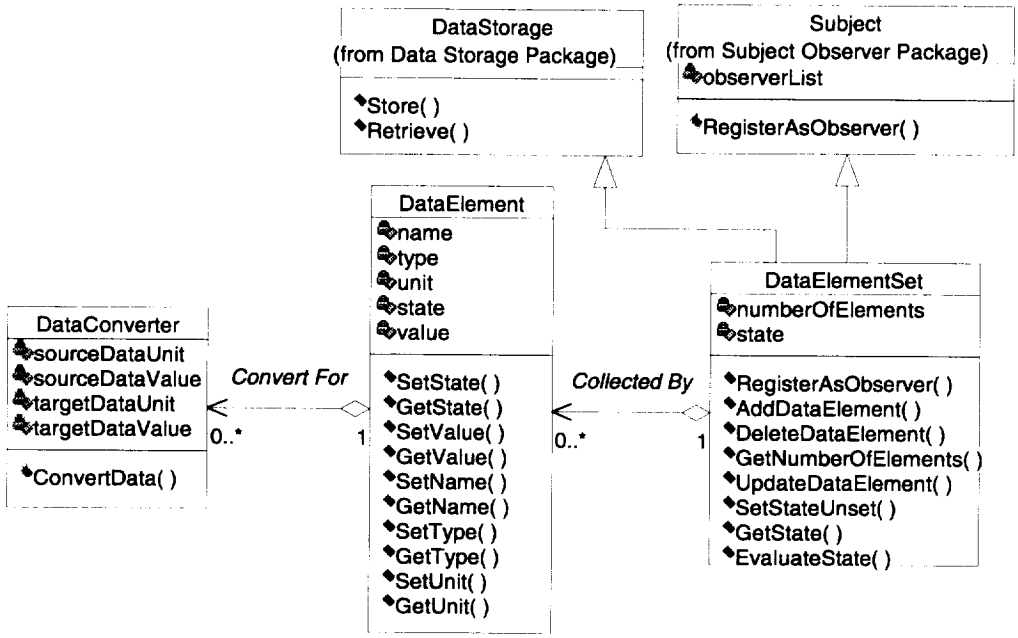
Private Properties	
macro	The macro attribute contains the macro used by the DataRelationship.
relationshipName	The relationshipName contains the name of the DataRelationship.
Public Methods	
Create ()	The Create operation is used to create the DataRelationshipSpecification.
GetMacro ()	The GetMacro operation is used to get the macro for the DataRelationship.
SetMacro ()	The SetMacro operation is used to set the macro for the DataRelationship.
GetRelationshipName ()	The GetRelationshipName operation is used to get the name of the DataRelationship.
SetRelationshipName ()	The SetRelationshipName operation is used to set the name of the DataRelationship.
Store ()	The Store operation is used to store the contents of the DataRelationshipSpecification.
Retrieve ()	The Retrieve operation is used to retrieve the contents of the DataRelationshipSpecification.

DATA ELEMENT PACKAGE

The Data Element package contains the associations between the DataElements, the DataConverter, and the DataElementSet. This class diagram can be shown in

Figure 5-20. The DataStorage and Subject classes are shown in this package to illustrate their relationship to the DataElementSet. These classes do not belong to this package. The class diagram illustrates that the DataElementSet contains zero to many DataElements. Each DataElement has zero to many DataConverters for conversion of their units.

Figure 5-20. Data Element Class Diagram



DataElementSet

The DataElementSet is a collection of instances of DataElement. A list of properties and methods for this class can be found in Table 5-23. The DataElementSet inherits from the Subject and DataStorage classes, so it also contains the properties and methods shown in Tables 5-14 and 5-26.

Table 5-23. Properties and Methods for DataElementSet Class

Private Properties	
numberOfElement	The number of DataElements contained in the DataElementSet.
State	The state attribute contains the state of the DataElementSet.
Public Methods	
RegisterAsObserver ()	The RegisterAsObserver operation is used to allow the subject to register as an observer of its input DataElementSet.
AddDataElement ()	The AddDataElement operation is used to add a DataElement to a DataElementSet.
DeleteDataElement ()	The DeleteDataElement operation is used to delete a DataElement from a DataElementSet.

Table 5-23. Properties and Methods for DataElementSet Class (Continued)

GetNumberOfElements ()	The GetNumberOfElements operation is used to get the number of elements in a DataElementSet.
UpdateDataElement () return	The UpdateDataElement operation is used to update values, units, etc. in a DataElementSet.
SetStateUnset () return	The SetStateUnset operation is used to set the DataElementSet state to Unset.
GetState () return	The GetState operation is used by an observer to get the state of the DataElementSet.
EvaluateState () return	The EvaluateState operation is used by the DataElementSet to evaluate its state.

DataElement

A DataElement is a container for a piece of data. A list of properties and methods for this class can be found in Table 5-24.

Table 5-24. Properties and Methods for DataElement Class

Private Properties	
name	The name attribute contains the DataElement name.
type	The type attribute contains the DataElement type, i.e., int, float, etc.
unit	The unit attribute contains the DataElement unit, i.e., feet, inches, etc.
state	The state attribute contains the DataElement state, i.e., set or unset.
value	The value attribute contains the DataElement value.
Public Methods	
SetState ()	The SetState operation is used to set the state of each DataElement.
GetState ()	The GetState operation is used to get the state of each DataElement.
SetValue ()	The SetValue operation is used to set the value of each DataElement.
GetValue ()	The GetValue operation is used to get the value of each DataElement.
SetName ()	The SetName operation is used to set the name of each DataElement.
GetName ()	The GetName operation is used to get the name of each DataElement.
SetType ()	The SetType operation is used to set the type of each DataElement.
SetUnit ()	The SetUnit operation is used to set the unit of each DataElement.
GetType ()	The GetType operation is used to get the type of each DataElement.
GetUnit ()	The GetUnit operation is used to get the unit of each DataElement.

DataConverter

The DataConverter converts the unit and value for a data element. A list of properties and methods for this class can be found in Table 5-25.

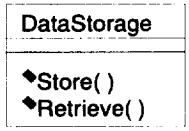
Table 5-25. Properties and Methods for DataConverter Class

Private Properties	
sourceDataUnit	The sourceDataUnit attribute defines the source DataElement's unit, i.e.,int, float, etc.
sourceDataValue	The sourceDataValue attribute defines the source DataElement's value.
targetDataUnit	The targetDataUnit attribute defines the unit for the target after the conversion.
targetDataValue	The targetDataValue attribute defines the value for the target after the conversion.
Public Methods	
ConvertData ()	The ConvertData operation is used to convert the unit and value of the DataElement.

DATA STORAGE PACKAGE

The Data Storage package contains the data storage/retrieval interface that enables local or global storage and retrieval. The class diagram is shown in Figure 5-21.

Figure 5-21. Data Storage Class Diagram



DataStorage

The DataStorage class manages storage and retrieval of data objects. A list of properties and methods for this class can be found in Table 5-26.

Table 5-26. Properties and Methods for DataStorage Class

Public Methods	
Store ()	The Store operation is used to store data objects.
Retrieve ()	The Retrieve operation is used to retrieve data objects.

DSSA Substage 4-7: Develop State Diagrams

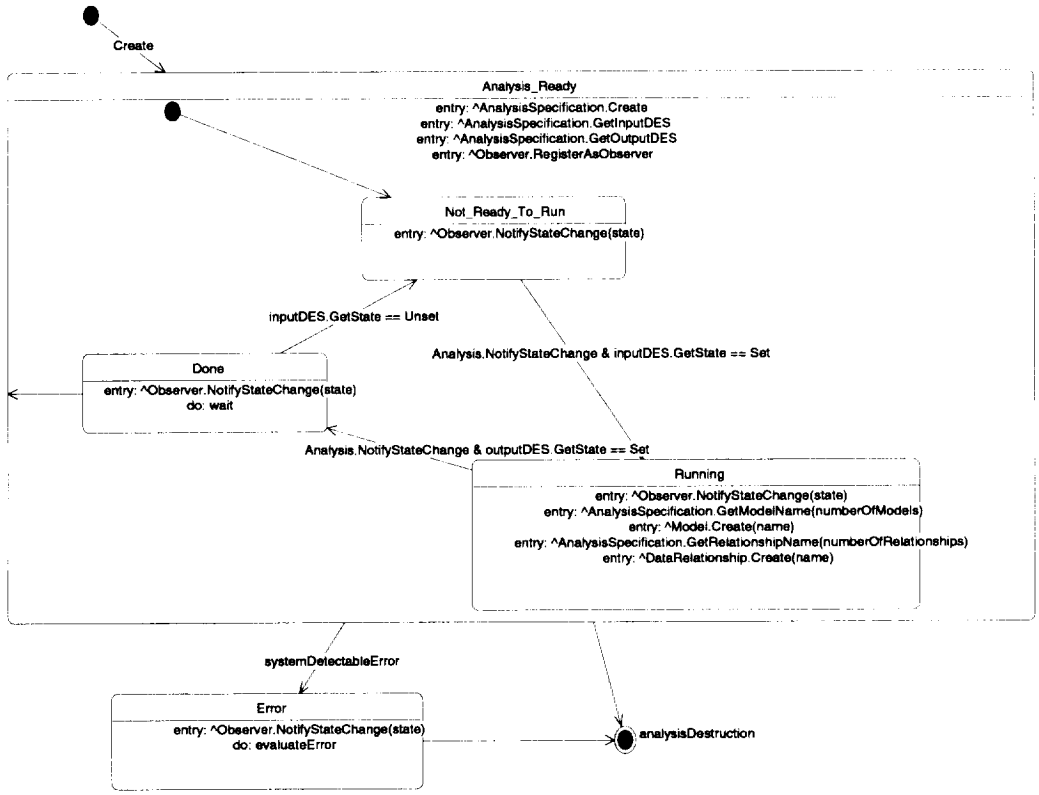
State diagrams describe all possible states of a particular object and how the object's state changes on particular events. The following sections contain state diagrams only for the classes that require states.

ANALYSIS STATE DIAGRAM

The Analysis has four states: "Not Ready To Run," "Running," "Done," and "Error." On creation of the Analysis, it creates an AnalysisSpecification, receives

input and output DataElementSets from the AnalysisSpecification, and registers as an observer to its input DataElementSet. At the same time, the initial state of the Analysis is set to the "Not Ready To Run" state. When the Analysis is notified of a state change on its input DataElementSet, the Analysis finds out what the state is. If the input DataElementSet is in the "Set" state, the Analysis will change to the "Running" state and notify its observers. In the "Running" state, it will create the Models and DataRelationships needed for the Analysis and will wait for its output DataElementSet to become "Set." Once the output DataElementSet of the Analysis is "Set," it will go to the "Done" state where it will remain until either the Analysis input DataElementSet becomes "Unset" (at which time the Analysis will go to the "Not Ready To Run" state), or it is destroyed. Upon a system error, the Analysis will go to the "Error" state. Figure 5-22 shows the Analysis state diagram.

Figure 5-22. Analysis State Diagram

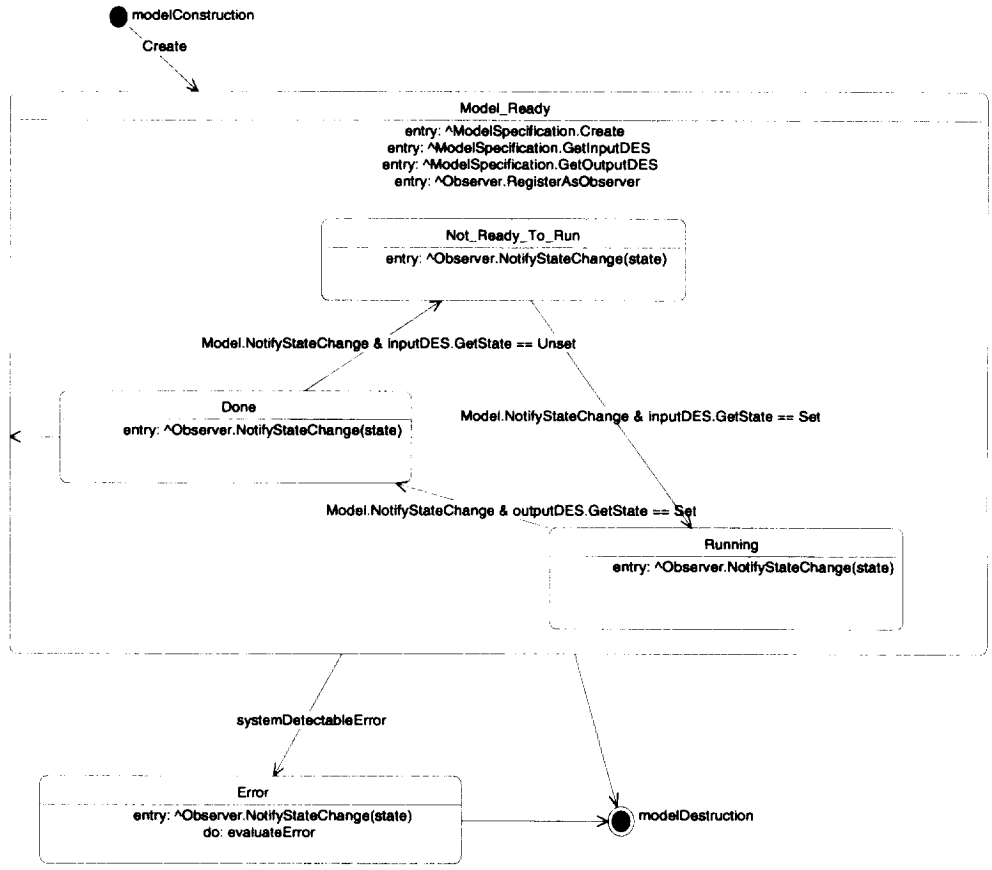


MODEL STATE DIAGRAM

The Model class has four states: "Not Ready To Run," "Running," "Done," and "Error." On creation of the Model, it creates a ModelSpecification, receives input and output DataElementSets from the ModelSpecification, and registers as an observer to the input DataElementSet. At the same time, the initial state of the Model is the "Not Ready To Run" state. When the Model's input DataElementSet goes to the "Set" state, the Model will change state to the "Running" state and

notify its observers. In the "Running" state, the Model will perform its transformation and will wait for its output DataElementSet to become "Set." Once the Model output DataElementSet is "Set," it will go to the "Done" state and will remain there until either the Model input DataElementSet becomes "Unset" (at which time the Model will go to the "Not Ready To Run" state), or it is destroyed. Upon a system error, the Model will go into the "Error" state. Figure 5-23 shows the Model state diagram.

Figure 5-23. Model State Diagram



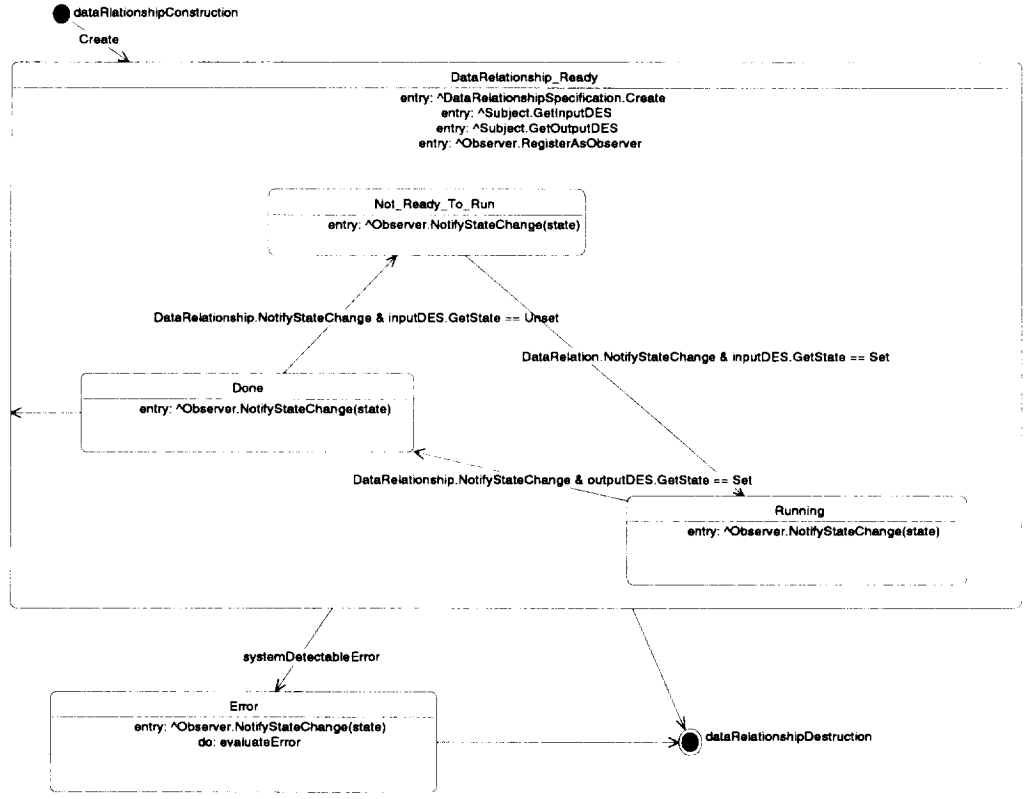
DATARELATIONSHIP STATE DIAGRAM

The DataRelationship class has four states: "Not Ready To Run," "Running," "Done," and "Error." On creation of the DataRelationship, it creates the DataRelationship Specification. The DataRelationship then receives input and output

DataElementSets from the DataRelationship Specification and registers as an observer to the input DataElementSet. At the same time, the initial state of the DataRelationship is the "Not Ready To Run" state. When its input DataElementSet goes to the "Set" state, the DataRelationship will change state to the "Running" state and notify its observers. In the "Running" state, the DataRelationship will perform its transformation and will wait for its output DataElementSet to become

"Set." Once the DataRelationship output DataElementSet is "Set," it will go to the "Done" state and will remain there until either the DataRelationship input DataElementSet becomes "Unset" (at which time the Model will go to the "Not Ready To Run" state), or it is destroyed. Figure 5-24 shows the DataRelationship state diagram.

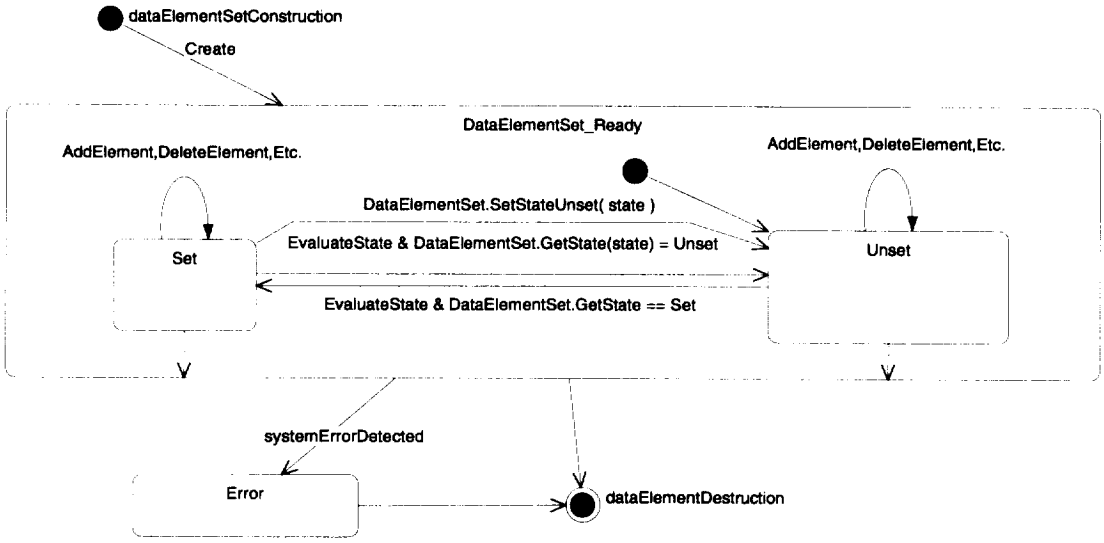
Figure 5-24. DataRelationship State Diagram



DATAELEMENTSET STATE DIAGRAM

The DataElementSet class has three states: "Set," "Unset," and "Error." On creation of the DataElementSet, its initial state will be the "Unset" state. The DataElementSet will go to the "Set" state when it evaluates its state and finds all of its DataElements are in the "Set" state. It can be forced to the unset state by its observers. Upon a system error, DataElementSet will go into the "Error" state. Figure 5-25 shows the DataElementSet state diagram.

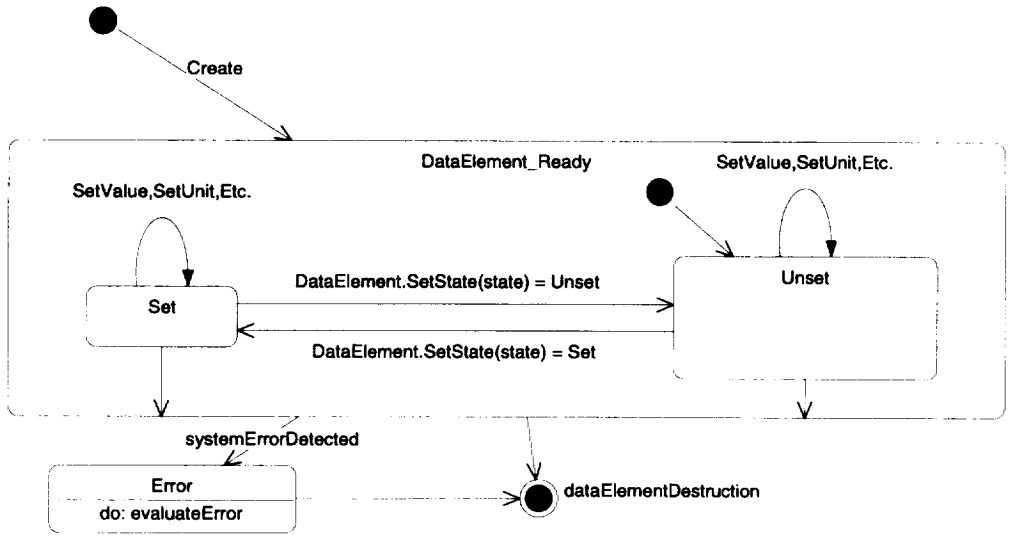
Figure 5-25. DataElementSet State Diagram



DATAELEMENT STATE DIAGRAM

The DataElement class has three states: "Set," "Unset" and "Error." On creation of the DataElement, its initial state will be the "Unset" state. The DataElementSet will go to the "Set" state when its state is changed to "Set" by the SetState() command. It will change to the "Unset" state when the SetState() command sets it to "Unset." Upon a system error, DataElement will go into the "Error" state. Figure 5-26 shows the DataElement state diagram.

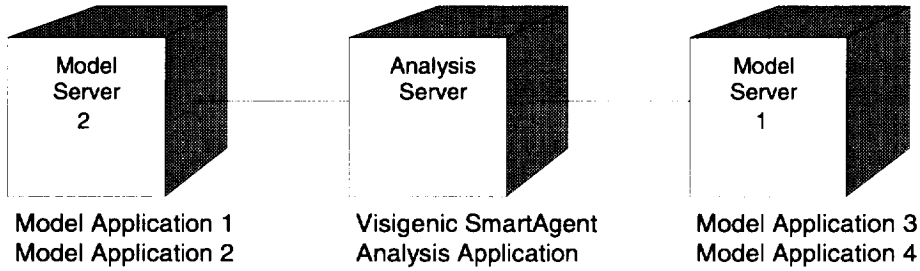
Figure 5-26. DataElement State Diagram



DSSA Substage 4-8: Develop Deployment Diagrams

A deployment diagram shows processors, devices, and their connections. A processor is a hardware component capable of executing programs, i.e., a computer. A device is a hardware component with no computing power, i.e., hardware controller or modem. There are no devices in the ASAC EA system, so the POC Deployment Diagram contains only processors and their connections with each other. The Deployment Diagram is shown in Figure 5-27. It is a generic model showing that there will be an Analysis Server with a Visigenic Smart Agent on it as well as multiple Model servers that will have Model Applications running on them.

Figure 5-27. POC Deployment Diagram



DSSA Substage 4-9: Review and Iterate

Review and iterate the items developed in DSSA stage 4.

DSSA STAGE 5—IDENTIFY REUSABLE ARTIFACTS

The goal for this phase of the domain-engineering process is to populate the software architecture high-level design(s) with components that may be used to generate new applications in the domain.

The following substages of DSSA stage 5 will be completed during the ASAC design effort:

- 5-1 Develop and collect the reusable artifacts
- 5-2 Develop each module
- 5-3 Requirements, verification, and testing
- 5-4 Review and iterate.

DSSA Substage 5-1: Develop and Collect the Reusable Artifacts

This substage addresses how to determine the best source of components to populate the software architecture. It is often referred to as the make, buy, or modify decision.

ASAC SERVICES

Three product types are necessary to provide the ASAC Services identified in the *ASAC Executive Assistant Architecture Description Summary*. They are

- message broker,
- binding language, and
- software distributor.

Additional product types researched included

- security services,
- directory services, and
- expert systems.

For the ASAC EA implementation, the security and directory services are provided by the message broker products, so additional security and directory service products are not required. A small development effort will provide the same functionality as an expert system, therefore, a stand-alone expert system is not required.

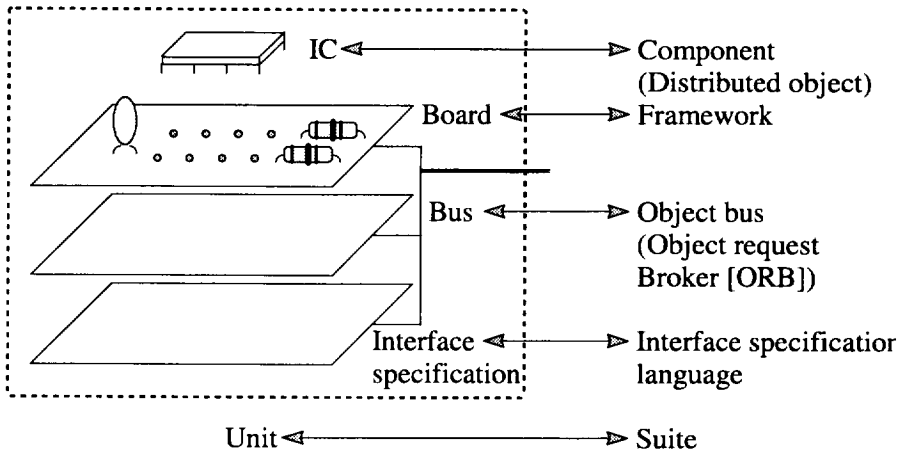
Message Broker Overview

A message broker is a software service layer that provides interoperability between software systems on different platforms. It represents the transaction and presentation layer of the ISO OSI model. It is generally used for communication between message brokers residing on different systems. It provides a heterogeneous view to software components residing on different platforms. For the ASAC EA system it will be the backbone of the EA services, and provide the infrastructure to tie in legacy systems and models in the existing models set, allowing a single model repository for distributed model development. It will provide the interface for client-server communications using a message-oriented architecture in an open environment.

There is a depiction of the software component model relative to a hardware model in Figure 5-28. A message broker is analogous to a hardware back panel, or bus. It provides the interface and the communication protocol to allow cards (components) to communicate, based on a predefined and deterministic protocol.

Like hardware, the cards can be produced by any manufacturer that conforms to the interface specified by the bus. This interoperability is what is achieved by message brokers.

Figure 5-28. Software Components (Distributed Objects)



Binding Language Overview

A binding language is software that provides an interface between the Analysis Application and the ASAC EA models. A binding language is needed to interface with the Model Application because the models were written without consideration for the calling conventions supported between supported models and legacy models. It provides a level of abstraction that ensures interoperability of the application across different interfaces.

Software Distributor Overview

A software distributor allows a server to automatically broadcast updates to numerous clients simultaneously at a scheduled time (push technology), or allows a client to selectively extract updates and install code from the appropriate server upon logging on (pull technology). This process enables the management of the whole software distribution process with minimal human intervention thereby reducing and/or removing the problems associated with version and update control.

Table 5-27 provides a mapping of ASAC services that each product type should provide, as well as services that will require substantial development. This mapping validates that each ASAC Service is provided by a product type or development effort.

Table 5-27. ASAC EA Services to Product Mapping

ASAC services	Product services	Object request broker	Binding language	Software distribution	Development
Distributed computing services					
Remote process service	ORB services	X			
Directory service	Naming service	X			
Data interchange service	ORB services	X			
Analysis service	ORB services	X			
Thread management service	ORB services	X			
Presentation services					
User interface service	Client binding language/browser		X		
Alert notification service	Client binding language/browser		X		
Data services					
Data administration service	API to Sybase	X			
Data management service	API to Sybase	X			
File input and output service	C++ I/O Streams		X		
Software distribution service	Electronic S/W distribution			X	
Catalog service	Catalog service				X
Management services					
Security administration service	Security service	X			
System administration service	API to HP-UX				X
Application management service	API to HP-UX				X
System management service	Transaction service	X			
Audit service	Audit service				X
Error management service	IDL exceptions	X			
Performance monitoring service	Load balancing service	X			
Security service	Security service	X			
Communication services	Communication services				
Communication management service	TCP/IP	X			
Network service	TCP/IP	X			
Intra-application communication service	IIOP	X			
Transaction management service	Transaction service	X			
Queuing service	ORB services	X			
Load balancing service	Load balancing service	X			
Common support services	Common support services				
Alerts service	IDL exceptions	X			
Message service	ORB services	X			
Help service	Browser				X

Support Services

Additional products to provide support services were identified. They include

- software libraries and
- software development tools.

Software Library Overview

A software library is a set of reusable software components, which are developed, fully tested, and are placed under configuration management control. Software libraries are typically developed internally or by a third party with the intention of being used directly in the development of the ASAC EA system.

Software Development Tool Overview

Software development tools are an integrated family of software products that enhance or aid in the development of an application design, modeling, coding, and testing. They provide iterative development, including visual modeling of system requirements, source-code generation, and reverse engineering of existing source code into a graphical object model.

SOFTWARE EVALUATION PROCESS

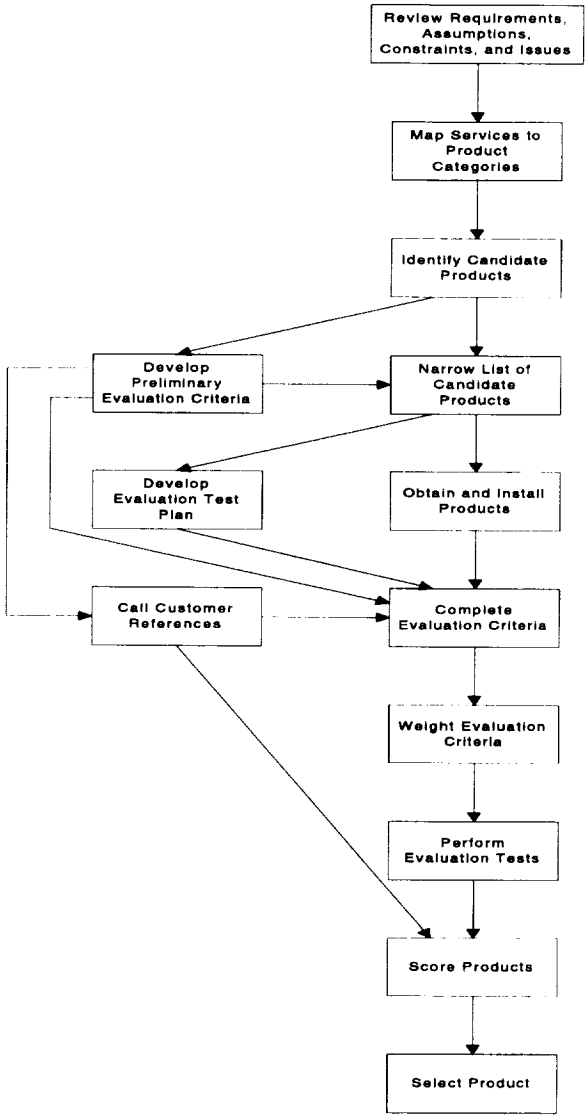
Each major software product will be evaluated according to a software evaluation process which is as follows:

- Review ASAC EA requirements, assumptions, constraints, and issues.
- Map ASAC EA services to product categories.
- Identify candidate products.
- Develop a preliminary list of evaluation criteria.
- Narrow the list of candidate products.
- Obtain and install evaluation copies of the candidate products.
- Develop an evaluation test plan.
- Call customer references.
- Complete the list of evaluation criteria.
- Assign weighting to the evaluation criteria.
- Perform evaluation tests.
- Score products against evaluation criteria.

- Select a product.

Figure 5-29 is a flowchart of the evaluation process.

Figure 5-29. Software Evaluation Process Flowchart



MESSAGE BROKER EVALUATION

Review ASAC EA Requirements, Assumptions, Constraints, and Issues

A review of the ASAC requirements, assumptions, constraints, and issues was performed to find the applicability to message brokers. This information was used to identify candidate products and develop a preliminary list of evaluation criteria.

Map ASAC EA Services to Product Categories

The services identified in the ASAC EA Products to Services Mapping (Table 5-27) were used to identify candidate products and develop a preliminary list of evaluation criteria.

Identify Candidate Products

This step involved the identification of potential message broker products. Industry journals, periodicals, books, and on-line data sources were used to identify candidate products.

Initial considerations of the client-server standards or models are as follows:

- Common Object Request Broker Architecture (CORBA)
- Distributed Component Object Model (DCOM)
- Distributed Computing Environment (DCE)
- Remote procedure calls (RPC)
- Message queuing
- Named pipe communication
- Other inter-process communication (IPC) mechanisms.

Common Object Request Broker Architecture

Based on distributed-object technology, CORBA is glue logic (middleware) that makes it possible for objects to dynamically discover and interact with one another across different platforms, operating systems, and networks. CORBA specifies an extensive set of services for creating and deleting objects, accessing them by name, storing them in persistent stores, externalizing their state, and defining ad hoc relationships between them.

Distributed Component Object Model

DCOM is a Microsoft standard for implementing distributed objects. It specifies interfaces between component objects within a single application or over a network. DCOM is designed for use across multiple network transport layers, including Internet protocols such as HTTP.

Distributed Computing Environment

DCE is an architecture consisting of standard programming interfaces, conventions, and server functionality (e.g., naming, distributed file system, remote procedure call) for distributing applications transparently across networks of

heterogeneous computers. DCE is promoted and controlled by the Open Software Foundation (OSF).

Remote Procedure Calls

RPC is a protocol that allows a program running on one host to execute a program on another host. RPC is an easy and popular paradigm for implementing the client-server model of distributed computing.

Message Queuing

Message queuing is a medium for passing messages between client and server applications using messages. A queue can be available dynamically in Random Access Memory or on permanent physical device.

Named Pipe Communication

Named pipe communication is a first-in first-out that facilitates one-way communication between two processes that are not necessarily spawned from the same parent.

Other IPC Mechanisms

Other IPC mechanisms include Shared Memory Segments, Transmission Control Protocol/Internet Protocol (TCP/IP) Sockets, and UDP/IP Datagrams. Shared Memory Segments allow multiple processes to share one common region of memory for the purpose of passing information. TCP/IP Sockets and UDP/IP Datagrams facilitate transparent interprocess communication across machines.

A decision was made to concentrate on products that are based on distributed-object technology since it promises the most flexible client-server systems. This is because distributed-object technology encapsulates data and business logic in objects that can roam anywhere on the network, run on different platforms, talk to legacy applications by way of object wrappers, and manage themselves and the resources they control. Software components designed as objects can be modified without affecting the rest of the components in the system or their interoperability. Out of several distributed-objects technologies, the two dominant industry implementations, CORBA and DCOM, were selected for evaluation in ASAC EA.

Six object request brokers were researched, one compliant to the DCOM standard and five compliant to the CORBA standard, as shown in Table 5-28. They are the following:

1. DAIS by ICL is a set of CORBA-based software tools to create and run a distributed application. The DAIS run-time libraries contain a CORBA compliant ORB. DAIS supports C and C++.

2. EntireBroker by Software AG provides a set of platform and transport independent communication services based on Microsoft's DCOM architecture for distributed objects, enabling a variety of applications to communicate together. Supported platforms include MVS, VSE, BS2000, UNIX, and Windows NT.

3. ObjectBroker by BEA Systems, Inc. (formerly called DEC ObjectBroker), is a CORBA 2.0 object request broker with full CORBA compliant C++ language bindings. It supports DCE's Generic Security Services API (GSSAPI), which allows the use of both DCE-based security (Kerberos) and other third-party authentication packages. ObjectBroker supports C and C++.

4. Orbix from IONA Technologies is a full and complete implementation of CORBA. Orbix runs on more than 20 operating systems with seamless operability guaranteed across all supported platforms. It provides support for C++, Java, Ada95, and Smalltalk.

5. PowerBroker CORBAplus by ExperSoft is a comprehensive implementation of the CORBA 2.0 specification. It includes asynchronous requests, multithreaded support, and delivers visual tools for editing the Interface Repository. PowerBroker CORBAplus is currently available for Windows 95/NT, Solaris, and HP-UX, and supports both C++ and Java.

6. VisiBroker by Visigenic Software, Inc. (formerly called ORBeline), is a CORBA 2.0 object request broker with a native implementation of the IIOP protocol. VisiBroker features an agent-based, multithreaded architecture with automatic configuration and smart binding. It also provides load balancing and high availability, enabling easy object migration and replication. VisiBroker supports both C++ and Java.

Table 5-28. Candidate Message Broker Products

Product	Vendor	Standard
DAIS	ICL PLC	CORBA
EntireBroker	Software AG	DCOM
ObjectBroker	BEA Systems, Inc.	CORBA
Orbix	IONA Technologies, PLC	CORBA
PowerBroker CORBA plus	ExperSoft Corporation	CORBA
VisiBroker	Visigenic Software, Inc.	CORBA

Once the products were identified, a dialogue was started with each vendor, product literature was requested, and technical briefings were attended. This material formed the basis for the initial analysis of the message broker products. Detailed vendor questionnaire responses can be found in Attachment A to this document.

Develop a Preliminary List of Evaluation Criteria

A preliminary list of evaluation criteria was derived based upon standard services based on requirements identified in the ASAC EA Architecture Description. Industry journals, periodicals, books, and on-line data sources, product literature, etc., were used to define additional items in the criteria based on similar problem domain and applicable studies. Nine major categories were developed for the evaluation. They are

1. Platforms Supported,
2. Language Bindings,
3. Standards Compliance,
4. Usability and Customizability,
5. Functional Features,
6. Development Issues,
7. Business Issues,
8. Market Acceptance, and
9. Cost and Training.

The evaluation categories are described in more detail below. The final evaluation criteria matrix is located in Appendix C.

Platforms Supported

Message broker components will potentially run on the following server and client platforms: HP-UX 10.20, Windows NT, SGI IRIX v5.3, Sun OS v5.4 or above, Windows95, Windows 3.1, Macintosh 7, Java Virtual Machine and AIX.

Language Bindings

Message broker components will potentially use the following language: C, C^{++}, and Java.

Standards Compliance

The most important issue in evaluating an ORB is its conformance to the CORBA 2.0 standard. Most commercial ORB's claim CORBA standard compliance, but the extent of the compliance may vary from product to product and vendor to vendor. CORBA 2.0 compliance is defined as adherence to the CORBA 2.0 standard; promises interoperability between independently developed applications across heterogeneous networks of computers. The CORBA 2.0 standard has been widely

adopted by 750 member companies. This extensive support provides some assurance of interoperability between CORBA ORBs and plug-and-play capability amongst IIOP based products. Note: DCOM was eliminated from our evaluation, so DCOM compliance is not discussed in this section. See "Narrow the List of Candidate Products" for more information.

CORBA Services are a part of the CORBA 2.0 standard. CORBA Services provide the capability to extend the basic operation of an ORB to support current and future application needs. The following 13 services have been defined by the OMG and have varying availability in commercial ORBs.

1. Life Cycle Service
2. Persistence Service
3. Naming Service
4. Event Service
5. Transaction Service
6. Concurrency Control Service
7. Relationship Service
8. Externalization Service
9. Security Service
10. Licensing Service
11. Object Query Service
12. Properties Service
13. Time Service.

Usability and Customizability

Usability and customizability are elements that can significantly enhance an ORB's utilization. Some of the issues to address regarding an ORB's usability and customizability are the following

- How easy is installation?
- How easy is configuration?
- How portable is the ORB implementation between platforms?

- How portable is the ORB implementation between different vendors?
- Is the ORB documentation accurate and detailed?
- How is the ORB performance compared to other ORBs?

Functional Features

Functional features were based on ASAC EA required services such as Load Balancing, Error Management, Interoperability, Fault-Resilience, and Audit Trail, that are not currently covered by the CORBA 2.0 standard services.

Development Issues

A friendly development environment can assist the developer in rapidly building the application. A sophisticated debugging capability can increase a developer's productivity in discovering and fixing logic problems.

Business Issues

When buying a commercial ORB product, technical support is very important. The vendor must provide support for the current product and strategic evolution of the product to meet the developing needs of the customers. The company's viability, quality of support, dedication to the product, and response to changing market needs may be as important as technical aspects of the product.

Market Acceptance

When defining the evaluation of market acceptance, the maturity of the product was considered, including features, as well as market share that can provide insight into user acceptance of the product.

Cost and Training

Like most buying decisions, cost is an important factor. Full life-cycle cost and the cost associated with the learning curve were researched.

Narrow the List of Candidate Products

There is an ongoing industry debate between the CORBA and DCOM users. Both have merits in their respective application domains, but CORBA seemed more attractive than DCOM for the following reasons:

- *Cross-platform support.* At present, the ActiveX/DCOM implementation is limited to Microsoft for Windows 95/Windows NT and UNIX (limited support) through BEA and Software AG. CORBA is a sponsored specification of 750-member vendors, including Microsoft and Digital. These

member companies support a wide variety of platforms including MS-DOS, 16-bit and 32-bit Microsoft Windows platforms, most UNIX implementation, OS/2, OS/400, Mac/OS, VME, MVS, VMS, and a number of real-time operating systems.

- *Cross-language support.* CORBA is designed to allow objects written in supported languages (i.e.,Java, C++) to be ported to different ORBs via an Interface Definition Language (IDL) that is CORBA standard, while hiding the underlying implementation of the ORB. This elevates the level of abstraction from the binary level to higher level language constructs. DCOM provides binary level interoperability, which is closely tied to a DCOM specific API, making portability of code problematic.

- *Architecture.* CORBA component/objects follow the classical Object Oriented (OO) model and supports multiple inheritance, encapsulation, and polymorphism. As of this writing, DCOM components are essentially black boxes, encapsulated OLE, which can not be extended via inheritance to create new instance components.

A high-level initial evaluation was performed to refine the list of candidates. In particular, the market presence and viability of the company, product maturity, stability of future product development and enhancement plans, and the availability and quality of technical support were assessed. Out of the six products being evaluated the following were eliminated:

BEA's ObjectBroker

This product was recently sold to BEA by DEC. A stable plan for future product releases, pricing, and support/upgrades were unobtainable from BEA.

ICL's DIAS

ICL is a European-based corporation with no technical support presence in the United States of America.

Software AG's Entire Broker

Did not support CORBA.

The ORB evaluation was continued with the following three products:

1. ExperSoft's PowerBroker
2. IONA's Orbix
3. Visigenic's VisiBroker.

Obtain and Install Evaluation Copies of the Candidate Products

Evaluation copies of each of the products listed in Table 5-29 were received and installed on the platforms listed below. Note that the server-operating environment is defined as HP-UX version 10.20, however, evaluation copies were not readily available for all platforms. Table 5-29 lists the products that were installed for evaluation and the platforms on which they were installed.

Table 5-29. Candidate Message Broker Product Install Platforms

Vendor	Product	Platform
ExpertSoft	PowerBroker	Windows 95
IONA	Orbix	Windows 95, HP-UX 10.20
Visigenic	VisiBroker	Windows 95, HP-UX 10.20

Develop an Evaluation Test Plan

An evaluation test plan was written to permit consistent investigation and evaluation of the message broker products. The plan described the product features to be tested and demonstrated. The plan called for the following activities to be performed

- Product installation
- Execution of a sample program within a machine
- Execution of a sample program across two local machines
- Execution of a sample program across two remote machines
- Implementations of security, load balancing, and directory services.

Call Customer References

ExperSoft and Visigenic supplied customer references. The references were users whose systems were representative of the ASAC EA. One reference for each vendor was contacted via telephone conference call and questioned about their use and experiences with the product. The following topics were discussed with each reference:

- Nature of the project (communication, finance, etc.)
- Size of project
- Overall architecture of project
- Duration of project

- Development environment (operating system, database, etc.)
- Runtime environment (if different than development)
- Other ORB products being considered and criteria for selection
- Percentage of ORB development compare to overall development
- Capabilities of ORB
- Utilization of ORB
- Ease of installation of ORB product
- Ease of development using ORB
- Problems encountered during design and development
- Ease of integration of ORB with other products
- Frequency of ORB product updates during project development
- Technical support response time
- Overall opinion of ORB.

The detailed questions and answers for each customer reference are located in Appendix C. Note: References for IONA were unavailable. Research was substituted for customer contact.

Complete the List of Evaluation Criteria

Knowledge obtained during research and testing enabled us to refine and complete the list of evaluation criteria. Evaluation criteria are used to select a product because

- the criteria enforces a rigorous and standard product comparison,
- they reduce the likelihood of new ideas surfacing post facto,
- they remove or surface biases,
- they provide a documentation trail, and
- they provide decision logic.

The evaluation criteria are divided into major categories, subcategories, and subcategory questions. Major categories are the highest level of criteria grouping; nine major categories were defined. Subcategories decompose major categories

into groups. Subcategory questions were used to evaluate a product in a given category. The number of questions per category varies. A total of 101 questions were defined for the evaluation criteria. The completed list of evaluation criteria can be found in Appendix C.

Assigning Weighting to the Evaluation Criteria

Weights were assigned to each major category based upon the category's importance to the ASAC EA. The total weight for the major categories is 100. After completing major category weighting, weights were assigned to each subcategory and subcategory question. The total weight for subcategories within a major category is 100. Likewise, the total weight for questions within a subcategory is 100. This method ensures each question is weighted according to its relative importance.

Finally, possible answers for each question were determined and assigned a score (each question could receive a minimum score of 0 and a maximum score of 5). It was important to define the possible answers to each question in advance and assign each possible answer a score. This allowed for consistent product scoring throughout the remainder of the product selection process, no matter how many people conducted the information gathering.

Table 5-30 shows examples of the type of question asked in each of the categories.

Table 5-30. Evaluation Questions by Category and Subcategory

Category	Subcategory	Question
Platforms supported	Server side Client side	HP-UX 10.20 or above? Java Virtual Machine?
Language bindings	Server side Client side	C, C++, Java? C, C++, Java?
Standards compliance	CORBA 2.0 Compliance	Does the product support the Internet Inter-ORB Protocol (IIOP)?
	CORBA Services	Does the product allow objects on the bus to locate other objects by name?
Usability and customizability	Ease of Installation	Does the product provide automated installation?
	Ease of configuration	Does the product provide some capability that allows additions or modifications to the infrastructure?
	Portability (same vendor)	Is the server-side code easily portable across platforms?
	Portability (different vendor)	Is the server-side code easily portable to different ORB products?
	Documentation	Is the documentation accurate and detailed?
	Performance	Using the custom test script, what is the average time for one client to call a server?

Table 5-30. Evaluation Questions by Category and Subcategory (Continued)

Category	Subcategory	Question
Functional features	Load balancing	Does the product support load balancing?
	Audit trail	Does the product provide audit capability?
	Error management	Does the product provide error handling capability?
	Interoperability	Does the product provide cross platforms independence?
	Fault-resilience	Does the product support fail-over (standby servers) for fault recovery?
Development issues	Development environment	How easy is development effort using this product?
	Debugging aids	Does the product provide method tracing at any granularity?
Business issues	Technical support	What level of technical support does the vendor provide?
	Viability of the vendor	Is the vendor well capitalized?
	Dedication of the vendor to the product	Does the vendor have an influential representation at OMG?
	Strategic alliance partners	Has the vendor formed alliances with significant third-party providers?
Market acceptance	Maturity	When was the product introduced?
	Market share	What is the current market share relative to competitors?
Cost and training	None	For a 5-server and 40-client ASAC EA system, what is the product suite price range?

The weighted and scored evaluation criteria and questions can be found in Appendix C.

Perform Evaluation Tests

The activities prescribed in the evaluation procedures were performed and proved to be invaluable because it

- provided a basis for assessing each product's administrative features,
- provided a basis for assessing each product's programming interface,
- identified undocumented features (both positive and negative) for each product,
- confirmed or denied vendor claims made in sales and technical material,
- provided a basis for assessing the adequacy of the supplied technical documentation, and
- provided a basis for examining each product in the operating environment.

The premise behind testing each of these products was to perform an evaluation that would provide feedback on the ease/difficulty with which an application could be taken from concept to implementation using each of the middleware evaluation candidates. In addition, performance and capabilities of each product were of particular interest.

An evaluation copy of each vendor's product was obtained and installed on the ASAC EA target platform(s) shown in Table 5-31. Each of the vendor's provided "sample" applications that were built (according to vendor documentation) and executed to prove that our installation and environment was correct. Then, the sample application "count" (see Appendix C) was chosen to be the representative program to be built and implemented using each of the vendor's products.

Table 5-31. Test Response Time (ms)

Vendor	C++	Java
ExperSoft	10.83	12.232
IONA	31.182	12.26
Visigenic	31.155	12.241

The only requirements that were considered when choosing a sample test application were as follows:

- The application had to have a client component and a server component.
- The middleware product would be used to product to facilitate machine-distributed client-server communication.
- The application would be simple enough to allow comparison across different vendor implementations, yet robust enough to prove the core requirements needed by ASAC EA with respect to distributed client-server communication.

The "count" program was implemented, executed, and benchmarked for performance across two remote machines. The results of this test are reflected in Table 5-31.

Score Products Against Evaluation Criteria

Each question was answered and scored for each product based on research, testing, and customer citations. A total for each product for each question was calculated.

$$(MajorCategoryWeight) \times (SubcategoryWeight) \times (QuestionWeight) \times (Score) \times (NormalizationFactor) = PointsReceivedforeachQuestion$$

Note: The normalization factor scales the scoring to a basis of 500 points.

$$\sum (points\ received\ for\ each\ question) = points\ received\ for\ a\ subcategory;$$

$$\sum (points\ received\ for\ each\ subcategory) = points\ received\ for\ a\ category;$$

$$\sum (points\ received\ for\ each\ category) = product\ score$$

The final criteria and scoring were analyzed to ensure reasonableness and non-redundancy. The completed Evaluation Criteria Matrix is located in Appendix C.

Select a Product

As outlined in "develop a preliminary list of evaluation criteria," there are nine major categories that were used in the product evaluation process. Table 5-32 summarizes the relative performance of the ExperSoft, IONA, and Visigenic categories.

Based upon the results compiled using the Evaluation Criteria Matrix, Visigenic placed first, IONA placed second, and ExperSoft placed third. Given the small margin between Visigenic and IONA, we felt either product would suffice. Visigenic was selected because

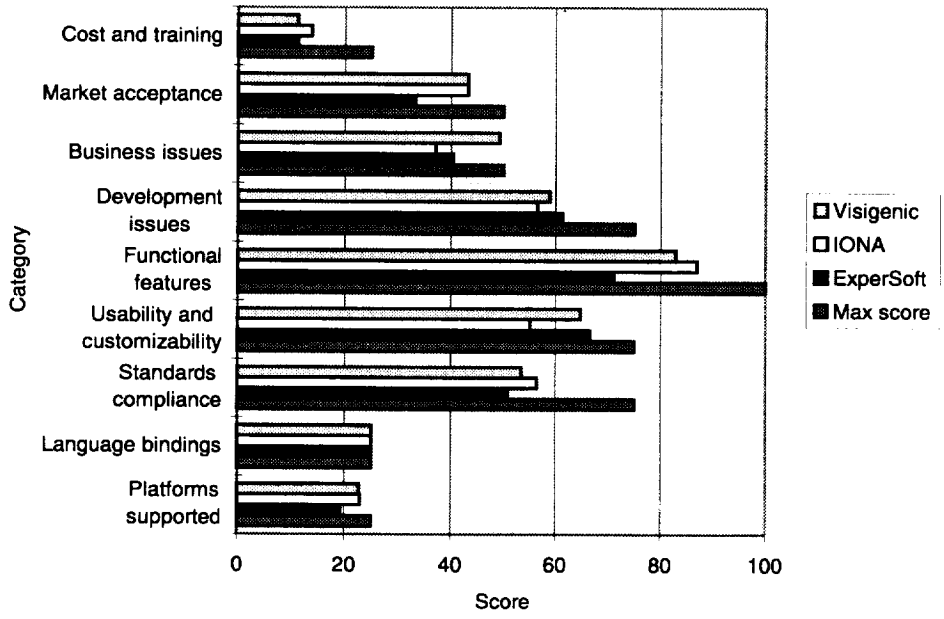
1. Visigenic's technology was adopted by major market leaders such as Netscape, Oracle, Novell, and Sybase;
2. Netscape made VisiBroker a part of Navigator™;
3. SunSoft collaborated with Visigenic on cross-platform support for Java, CORBA, and RMI conversion standards;
4. Visigenic offers the only native implementation of IIOP; and
5. Visigenic 5 offered timely technical support.

Table 5-32. Message Broker Evaluation Summary

Category	Max score	ExperSoft	IONA	Visigenic
Platforms supported	25	19.25	22.90	22.75
Language bindings	25	25.00	25.00	25.00
Standards compliance	75	50.93	56.55	53.55
Usability and customizability	75	66.53	55.13	64.80
Functional features	100	71.00	87.00	83.00
Development issues	75	61.35	56.55	58.95
Business issues	50	40.40	37.10	49.25
Market acceptance	50	33.25	43.25	43.25
Cost and training	25	11.25	13.75	11.25
Totals	500	378.96	397.23	411.80

Figure 5-30 is a graph that indicates how the three final products fared in the various categories. The graph depicts the actual scores received by each product in each major category. The maximum potential score for each major category is also shown on the graph.

Figure 5-30. Product Comparison by Category



BINDING LANGUAGE EVALUATION

Java and C++ were considered because they are the most pervasive languages in the industry today. Market acceptance and vendor support has made these two languages the "language of choice."

Java was chosen for the client and C++ for the server because of the tools available and the platforms designated for the client (Window/PC) and Server (Unix/Workstation).

SOFTWARE DISTRIBUTOR EVALUATION

Pull and push technologies were considered and evaluated based on the users of the ASAC EA system, as well as types of updates and frequency of the updates. A combination of both technologies may be used for updating software in the ASAC EA system. Effectively, "pull" technology may be used for having the client-side browser download the latest application from the server, upon login. "Push" technology may be incorporated for updating local stores of catalogs, model database information, and system analyses.

SOFTWARE LIBRARY EVALUATION

Third-party libraries were identified for both the client and server, as the development environments and primary development language for both sides are different. The client-side libraries are Java-based, therefore the following Java libraries have been identified: Java Reusable Components that are standard with Symantec Visual Café development environment. The server-side libraries are C++ based, therefore the following C++ libraries have been identified: Rogue-Wave C++ libraries; including standard, mathematical, and database; ANSI C++ runtime libraries. These software libraries were chosen based on market acceptance and favorable product reviews.

SOFTWARE DEVELOPMENT TOOL EVALUATION

Products identified are Symantec Visual Café for object development on the client, Rational Rose UML for overall design and C++ on HP-UX for object development on the server. CORBA IDL will be used to specify interface and interoperability. These development tools were chosen based on market acceptance and favorable product reviews.

DSSA Substage 5-2: Develop Each Module

Concurrent with product investigation and evaluation, the ASAC Design models are being built using the Rational Rose CASE tool. Initially, a proof of concept system is being produced that will be a running application that proves and demonstrates the more complicated concepts of the ASAC EA system. In particular, the key server modules for executing and running analyses and models in a distributed environment are being implemented with an integrated middleware product (Visigenic). The high-level interfaces to the remainder of the system are well-defined such that the system can be extended at a later date to include the remaining applications and integration to the legacy models. Taking this approach allows us to minimize risk by implementing the more difficult core pieces of the system up front.

DSSA Substage 5-3: Requirements, Verification, and Testing

Requirements, verification, and testing will be done as part of development in fiscal year 1998.

DSSA Substage 5-4: Review and Iterate

Review and iterate the items developed in DSSA stage 5.

Chapter 6
Conclusion

We used published and respected methodologies to define a design for the ASAC EA POC system. The design includes CRC cards, a role play script, a Use Case diagram, Interaction (Sequence and Collaboration) diagrams, Package diagrams, Class diagrams, State diagrams, and Deployment diagrams.

We also used a detailed evaluation process to start an Object Request Broker for use in the ASAC EA system. Additionally, we selected binding languages, software libraries, and development tools.

Work will begin on the next phase, development of the ASAC EA, in fiscal year 1998.

BIBLIOGRAPHY

Avellino, Ken. "Introduction to Client/Server Computing," Learning Tree International, 369/CN(2)/D.2/601/D.1, 1996.

Baer, Tony. "A Rational Approach to Objects" Software Magazine, November 1996.

Baer, Tony. "Components of Success?" Software Magazine, June 1997.

Bellin, David and Susan Suchman Simone. "The CRC Card Book," Addison-Wesley, 1997.

Booker, Ellis. "NASA Center Offers Formula for Picking Middleware," Web Week, May 12, 1997.

Carando, P. (1996a). An Object Request Broker Evaluation Instrument (white paper No. 121). AT&T Solutions.

Carando, P. (1996b). Selecting an Object Request Broker for Large-Enterprise, Distributed Computing (white paper No. 119). AT&T Solutions.

Carando, Patricia. "Selecting an Object Request Broker for Large-Enterprise," AT&T Solutions, 1996.

Carr, David F. "Object Wars," Internet World, February 1997.

Carr, David F. "An Object Sharing Standard that would Yield Net Harmony is Underminded by all Vendors Self-interest," Internet World Magazine, February 1997.

Carr, David F. "CORBA Application Architecture Is Outlined, VisiBroker 3.0 Fills Security Holes in Distributed Apps" Web Week, July 14, 1997.

Carr, David F. "Merging of Distributed Object Technologies" Web Week, July 14, 1997.

Clark, Scott. "File I/O with Jave: It Can Be Done!" Web Developer, May/June 1997.

Common Object Request Broker Architecture, OMG, July, 1995.

Common Object Services Specification, OMG, March, 1995.

CORBA (Common Object Request Broker Architecture) and the OMG (Object Management Group), http://www.acl.lanl.gov/CORBA/, February 1997.

CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments, IEEE Communications Magazine, Vol. 14, No. 2, February, 1997.

CORBAnet and the ORB Interoperability Showcase, http://www.omg.org/news/corbanet.htm, February 1997

CORBAServices: Common Object Services Specification, Vol. 1, March 1995.

Core, Gael. "Pushing Software Limits," Software Magazine, June 1997.

Dickman, Alan. "Race between Corba and ActiveX/Dcom," Information Week, January 20, 1997, Issue: 614.

Dickman, Alan. "Race between Corba and ActiveX/Dcom," Information Week, January 20, 1997, Issue: 614.

Domain Specific Software Architectures (DSSA), http://www.sei.cmu.edu/arpa/evo/dssa-sum.html.

Evolutionary Software Development, http://www.sei.cmu.edu/arpa/evo.html.

Fowler, Martin, and Kendall Scott. "UML Distilled—Applying the Standard Object Modeling Language," Addison-Wesley, 1997.

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. "Design Patterns—Elements of Reusable Object-Oriented Software," Addison-Wesley, 1995.

Gaudin, Sharon. "Cheveron is CORBA's biggest win; will link users to seismic data," ComputerWorld Magazine, April 1997.

Gill, Philip J. "The Java Revolution at Two Years," Object Magazine, June 1997.

Larson, Don. "Putting Java in its Place," Web Developer, May/June 1997.

Lockheed Martin Advanced Concepts Center and Rational Software Corporation. "Succeeding with the Booch and OMT Methods, A Practical Approach," Addison Wesley, 1996.

McCown, Michael and Jason Pritchard. "Corba Connection—Object protocol lets companies mix and match Corba ORBs," Information Week, March 17, 1997, Issue: 622.

Message Oriented Middleware Association, "Message Oriented Middleware Glossary," http://www.moma-inc.org.

Meta Group, Software AG, "Internet, Networking, and Middleware Glossary," 1996.

Millikin, Michael. "Distributed Objects: A New Model For The Enterprise," Data Communication Magazine, February 1997.

Mowbray, Thomas J. and Ron Zahavi. "The Essential CORBA," Wiley, 1995.

Newman, David S. "Systems Management for the Distributed Object Environment" http://www.technium-inc.com/osmpart2.html.

Orfali, Robert, and Dan Harkey. "Client/Server Programming with Java and CORBA," Wiley, 1997.

Orfali, Robert, Dan Harkey, and Jeri Edwards. "Instant CORBA," Wiley, 1997.

Orfali, Robert, Dan Harkey, and Jeri Edwards. "The Essential Client/Server Survival Guide," Wiley, 1996.

Orfali, Robert, Harkey, Dan, and Jeri Edwards. "The Essential Distributed Objects Survival Guide," Wiley, 1996.

Radcliff, Deborah. "Sybase Breaks from the Pack," Software Magazine, June 1997.

Rational Software Corporation UML Resource Center, "UML Document Set Version 1.1," September 1997, http://www.rational.com/uml/references/.

Roberts, Eileen, and James A. Villani. "ASAC Executive Assistant Architecture Description Summary," NASA Contractor Report 201681, April 1997.

Rumbaugh, James, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. "Object-Oriented Modeling and Design," Prentice Hall, 1991.

Savetz, Kevin M. "ActiveX vs. Java," Web Developer, January/February 1997.

Tracz, W. "Domain-Specific Software Architecture (DSSA) Frequently Asked Questions (FAQ), Version 1.2 - ADAGE-IBM-93-12B." Loral Federal Systems, Owego, March 1995. http://www.sei.cmu.edu/arpa/dssa/dssa-adage/faq/faq.html.

Tracz, W. "What is DSSA?" Loral Federal Systems, Owego. http://www.owego.com/dssa/what-is-dssa.html.

Tracz, W. and L. Coglianese. "Domain-Specific Software Architecture Engineering Process Guidelines, ADAGE-IBM-92-02B." Loral Federal Systems, Owego, 1992.

Tristram, Claire. "Middleware makes C/S apps really work," Datamation Magazine, August 1996.

Tucker, Michael Jay. "Bridge your legacy systems to the Web," Datamation, March 1997.

Whiting, Rick. "How IT Works—Push Technology," Client/Server Computing, June 1997.

Willett, Shawn. "Alliance to Tackle CORBA Drawbacks," Computer Reseller News, March 17, 1997, Issue: 727.

Wingrove, E. R. III, P. F. Kostiuk, R.C. Sickles, and D. H. Good. "The ASAC Air Carrier Investment Model (Revised), NS301RD2." Logistics Management Institute, June 1996.

Appendix A
Acronyms

ACSYNT	Aircraft Synthesis Model
AIA	Aerospace Industries Association
AND	Approximate Network Delay Model
API	Application Program Interface
APP	Application Portability Profile
ASAC	Aviation System Analysis Capability
AST	Advanced Subsonic Technology program
ATA	Air Transportation Association
ATC	Air Traffic Control
ATM	Automated Teller Machine
CAASD	Center for Advanced Aviation System Development
CASE	Computer Aided Software Engineering
CGI	Common Gateway Interface
COE	Common Operating Environment
CORBA	Common Object Request Broker Architecture
COSTMOD	Delay Cost Model
CRC	Class-Responsibility-Collaboration
CUA	Common User Access
DARPA	Defense Advanced Research Projects Agency
DBMS	Database Management System
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DES	DataElementSet
DOT	Department of Transportation
DSSA	Domain-Specific Software Architecture
EA	Executive Assistant
FAA	Federal Aviation Administration

FIFO	First In First Out
FLOPS	Flight Optimization System Model
GSSAPI	Generic Security Services API
GUI	Graphical User Interface
HP	Hewlett-Packard
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IIOP	Internet Inter-ORB Protocol
I/O	Input/Output
IDL	Interface Definition Language
IPC	Inter-process Communication
JSIMS	U.S. Army's Joint Simulation System
LAN	Local Area Network
LMI	Logistics Management Institute
NARIM	National Airspace Research and Investment Model
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
OMG	Object Management Group
OMT	Object Modeling Technique
OOD	object oriented design
ORB	object request broker
OSE	Open System Environment
OSF	Open Software Foundation
POC	Proof of Concept
QRS	Quick Response System
RPC	remote procedure call
SGI	Silicon Graphics, Incorporated
SQL	Structured Query Language
STAT	Small Transportation Aircraft Technology
S/W	Software
TBD	to be determined
TCP/IP	Transmission Control Protocol/Internet Protocol

UML	Unified Modeling Language
UDP/IP	User Datagram Protocol/Internet Protocol
WAN	Wide Area Network
WWW	World Wide Web

Appendix B
Domain Dictionary

Alert Notification Service—a component of the Presentation Service that provides presentation for user- and application-generated alerts and notifications.

Alerts Service—a component of the Common Support Service that provides functionality for user- and application-generated alerts and notifications. It routes and manages alert messages throughout the domain.

Analysis—1. The unit of work for an analyst using the Executive Assistant.
2. A class that manages the creation and instantiation of models and their relationships.

Analysis Application—a software component that interacts with user input, models, and drivers to create analytical results.

Analysis Data—the part of an analysis document that contains the input and output data from each stage of an analysis.

Analysis Document—a document that contains the analysis specification and analysis data for a specific analysis; it is an instantiation of an analysis template.

Analysis (Broker) Service—a component of the Distributed Computing Service that allows applications to use methods or objects that are remote. It allows objects to dynamically discover each other and interact across machines, operating systems, and networks.

Analysis Specification—manages specification data for a particular analysis.

Analysis Template—a generic document that provides an outline for performing specific types of analysis; the user customizes the template for a specific task and saves it as an analysis document.

Analyst—a person who interacts with the executive assistant to perform an analysis, define an analysis, save data, halt an analysis, inquire about the state of a running analysis, choose an analysis template, or choose an existing analysis to run.

Application Architecture—the architecture for a single system (the result of instantiating and refining a reference architecture).

Application Engineering—the process of instantiating/refining and/or extending a reference architecture.

Application Management Service—a component of the Management Service that maintains a dynamic configuration of application services. It starts the appropriate application services on appropriate machines. It also monitors application services to ensure they are available and performing correctly, restarts lost services, and starts and stops services.

Audit Service—a component of the Management Service that provides a permanent record of system usage. It includes user access, model usage, error logging, and time stamping.

Authentication—verification of the user's validity.

Authorization—control of user access.

Aviation System Analysis Capability (ASAC)—a decision support system consisting of models, databases, and tools used to support analysis of the effects of advanced technologies on the integrated aviation system.

Backsolver—a type of driver that finds a value for one or more variables, given a set of constraints.

Catalog Repository—database containing the registrations for all models and drivers within the Executive Assistant.

Catalog Service—a component of the Data Service that registers and assembles model and driver descriptions, configurations, I/O, type, and average execution time requirements.

Checkpoint—a point along an analysis string at which the user chooses to suspend the analysis to examine its current state. Setting checkpoints lets the user stop an analysis run in between running models. At such a point, the user may view data both after it exits a model and before it enters the next (there may be differences because of units translation).

Class—A template for an object containing variables and methods representing behavior and attributes.

Class Library—A set of client programming tools. These tools can be used in a Java program or Web page-embeddable lava applet.

Client—a software program used to contact and obtain data from a server software program.

Client-Server—a relationship between two processes in which one makes a request to the other. It is possible for the server to in turn make a request to the client and thereby reverse the roles.

Common Support Service—a service that provides common system-level support across all components of the EA system.

Communication Management Service—a component of the Communication Service that provides overarching communication services between system applications and other services.

Communication Service—a service that provides communication among all components within the domain. It provides translation if more than one protocol is used. It also provides facilities for receiving data external to the domain and sending data out of the domain.

Data Access and Connectivity Service—a service that provides DBMS connectivity; performs query transactions.

Data Administration Service—a component of the Data Service that allows a System Administrator to enter, maintain, change, and remove data in repositories.

Data Flow Diagram—a graph on which nodes are processes and arcs are data flows; part of a functional model.

Data Interchange Service—a component of the Distributed Computing Service that provides specialized support for applications that may be dispersed among computer systems in the network but must maintain a cooperative processing environment.

Data Management Service—a component of the Data Service that provides DBMS connectivity and performs query transactions.

Data Service—a service that provides data administration, management, input/output, and distribution services.

Data Value—the actual value for some piece of data that is input or output of an analysis or model.

DataConverter—defines a common interface for converting a data value from one unit to another.

DataElement—a container for a piece of data.

DataElementSet—a collection of instances of DataElement.

DataRelationship—a class that observes a data source for changes in state. Gets data values from data source (when it is in particular state), applies a macro function to the data source, and sets the values in a data target.

DataRelationshipSpecification—manages specification data for a particular DataRelationship.

DataStorage—manages storage and retrieval of data objects.

DataTransformer—defines a interface for a class that transforms input data values into output data values.

Default Data—predefined input values for models.

Default Template—a template that contains no model or data.

Dependency Repository—a database containing the dependencies for all models and drivers within the Executive Assistant.

Directory Service—a component of the Distributed Computing Service that maintains a dynamic list of all application services, models, and drivers supported by EA.

Distributed Computing Service—a service that provides specialized support for applications that may be dispersed among computer systems in the network but must maintain a cooperative processing environment.

Domain Engineering—the process of creating a DSSA (Domain Analysis and Domain Modeling followed by creating a software architecture and populating it with components).

Domain Expert—an individual whose expertise and experience in the domain can lend insight into various aspects of the domain.

Domain Model—any representation of elements in a domain that shows some relationship among them. In DSSA this model usually consists of a lexicon, ontology, and taxonomy of the terms that characterize the domain, including objects, relationships, products, and perhaps behavioral terms such as actions and events.

Domain-Specific Software Architecture—a software architecture with reference requirements and domain model, infrastructure to support it, and process to instantiate and refine it.

Driver—an interactive problem-solving software application that interfaces with the analysis application; examples include an optimizer, backsolver, and table generator.

Driver Application—an application that handles different types of drivers.

Driver Developer—a person who develops a driver.

Driver-wrapped Model—a model or set of models that a driver uses to perform an analysis.

Dynamic Model—a model that describes the aspects of a system that change over time; used to specify and implement the control aspects of a system.

Error Management Service—a component of the Management Service that provides error management for system level errors. It preserves the integrity of the system.

Event—the occurrence of a condition, state change, or the availability of some information, that is of interest to one or more modules.

Event Trace Diagram—a diagram that depicts a sequence of events and the objects exchanging the events; part of a dynamic model.

Execution Point—a point along an analysis string at which the user chooses to perform an action; examples are a checkpoint, start point, and stop point.

Executive Assistant—subdomain of ASAC that encompasses applications with which an analyst can use a series of one or more integrated models and drivers to perform an analysis.

Extendability—the ability to add additional (new) functionality to a system.

File I/O Service—a component of the Data Service that provides file management capability.

Functional Model—a model that describes the data value transformations within a system; contains data flow diagrams.

Global—components that are available to all users.

Help Service—a component of the Common Support Service that provides help to users.

History Document—a document that contains the results of running an analysis using an analysis document.

Horizontal Growth—horizontal expansion of a system, e.g., 1 to 2 to 3 servers.

Inheritance—a feature of object-oriented programming subclasses that inherit the non-private variables and methods of all their superclasses. An object implicitly contains all the non-private variables of its superclass and can invoke all the non-private methods of its superclass.

Instance—an object. When a class produces an object, the object is an instance of the class.

Interface Definition Language (IDL)—the language developed by OMG to define a component's interfaces with potential clients. It provides operating system- and programming language-independent interfaces to all the services and components that reside on a CORBA bus.

Intra-Application Communication Service—a component of the Communication Service that provides communication capability among components within the EA system.

Java—an object-oriented programming language modeled after C++. It is designed to be small, simple, and portable across platforms and operating systems.

Load Balancing—a feature of the communications service that supports the ability of distributed applications to spread their work-loads amongst two or more duplicated applications components.

Load Balancing Service—a component of the Communication Service that provides load balancing among replicated models to ensure no model is overloaded.

Local—components that are local to a user's system; not available to all users.

Login—the account name used to gain access to a computer system.

Log in—to enter a computer system.

Logon Transparency—the use of a single password to gain access to all servers and their services within a system (multiple system elements recognize one log in).

Management Service—a service that provides system, application, error, performance, and security management.

Message—a bit of code sent between applications usually including some instructions for the receiving application.

Message Service—a component of the Common Support Service that handles message traffic (parsing and distribution) among applications. It provides message queuing capability.

Model—represents the interface to and state of a model application.

Model—a standalone software application that provides non-interactive data transformation.

Model Application—an application that handles different types of models.

Model Configuration—an execution path through a model that has unique inputs and outputs; every model has at least one configuration.

Model Developer—a person who develops a model.

Model Option—a mechanism that specifies the configuration of a model.

Model Specification—contains information about the data used to execute a model.

Network Service—a component of the Communication Service that provides a connection between the external communications network and the EA system.

Nominal Driver—the default driver that does not perform a function, e.g., not an optimizer, backsolver, or table generator.

Object—a concrete instance of some class.

Object Diagram—a graph on which nodes are object classes and arcs are relationships among classes; part of an object model.

Object Model—a model that describes the static structure of the objects in a system and their relationships; contains object diagrams.

Observer—a class that defines an updating interface for objects that should be notified of changes in a subject. In response to notification, observers query the subject to synchronize its state with the subject's state.

Optimizer—a type of driver that finds a minimum or maximum value for a variable.

Performance Monitoring Service—a component of the Management Service that provides routine diagnostics and performance monitoring.

Presentation Service—a service that provides the user interface layer. Here, the user formulates a request and receives a reply. This service also displays alert information to the user.

Queues—a simple data structure for managing the time-staged delivery of requests to servers. Queued elements may be sorted in some order of priority. Clients insert items in the queue and servers remove items from the queue, either as soon as possible, or in batch or periodically.

Queuing Service—a component of the Communication Service that provides process queuing to ensure fair access to system resources.

Reference Architecture—a software architecture for a family of application systems.

Registration—a description and specification of a model or driver; it is stored in the Catalog Repository and used by the Catalog Service.

Remote Process Service—a component of the Distributed Computing Service that provides remote procedure call (RPC) capability to the system.

Replication—hosting of multiple copies of an application or service on multiple machines to ease network contention and processor load.

Repository—a database that contains the catalog, template, and dependency repositories.

Scalability—the ability to add more of an existing function to a system.

Scheduling Service—provides thread management, queuing, and load balancing.

Security Administration Service—a component of the Management Service that maintains a registry of all authorized users throughout the domain and tracks which functions each user or group is allowed to perform (authorization). It assigns password protection, model security, groups, and permissions.

Security Service—a component of the Management Service that provides a single login service for all systems throughout the domain. It authenticates a user and provides logon transparency.

Server—a software program that provides a specific service to client software. A single server machine can run several different server software programs at the same time.

Software Distribution Service—a component of the Data Service that provides electronic software distribution capability.

Start Point—the point along an analysis string at which a user chooses to begin an analysis.

State Diagram—a graph whose nodes are states and whose arcs are transitions among states caused by events; part of a dynamic model.

State Observer—observes and records state information about its subject.

Stop Point—the point along an analysis string at which a user chooses to stop an analysis.

Subject—a class that defines the properties of an object being observed. A subject may have any number of dependent observers. All observers are notified when the subject undergoes a change in state.

System Administration Service—a component of the Management Service that provides the capability to configure, operate, maintain, and manage the local configuration.

System Administrator—person who administers configuration of Executive Assistant applications, adds users, administers passwords, administers security levels, and creates and administers model and driver registrations.

System Management Service—a component of the Management Service that provides system management and query governing and handles runaway queries.

Table Generator—a type of driver that populates a table of variables with values.

Template Developer—a person who develops a template.

Template Repository—database containing globally available templates.

Thread—a unit of concurrency provided in a program. They are used to create a programs execution environment that weaves together instructions from multiple independent execution paths or "threads."

Thread Management Service—a component of the Distributed Computing Service that provides thread management capabilities for applicable application services.

Transaction Management Service—a component of the Communication Service that supports creation and management of transaction logs and transaction processing.

User Application—software component with which the user interacts with the Executive Assistant.

User Interface Service—a component of the Presentation Service that provides direct interaction with a user through windows, icons, menus, keyboard, mouse, or other means.

Vertical Growth—vertical expansion, e.g., add additional models or clients to a system.

Web Browser—software that interprets the HTML commands and displays the page contents to a client.

Appendix C
Message Broker Evaluation Supporting Documentation

This appendix contains the following documentation:

- Customer questionnaire
- Evaluation test plan
- Evaluation test procedures
- Evaluation criteria matrix.

CUSTOMER QUESTIONNAIRE

We spoke with customers of ExperSoft and Visigenic to ask them questions about the products under evaluation. We were not able to contact customers from IONA.

The customer references and completed questionnaires for ExperSoft and Visigenic are provided below.

ExperSoft

Response to Customer Questionnaire from Vlad Kroutik, Sapient Inc, Tel (617) 621-0200, email vkrout@sapient.com.

PROJECT-ORIENTED QUESTIONS

Q. For what project did you use this product?

A. Used PowerBroker with 3 client projects.

Q. What was the nature of the project (communication, finance, etc.)?

A. A database-oriented project.

Q. What was the size of the project?

A. About 20 servers and 95 clients.

Q. How did the product fit in the project overall architecture?

A. Utilize CORBA as the middleware architecture to connect these analysis servers together.

Q. How long was the project?

A. Started a year ago.

Q. What was your development environment (operating system, database, etc.)?

A. Servers running HP-UX 10.20 and Windows 95 clients.

Q. What was your runtime environment (if different than development)?

A. Same as development.

Q. Did you consider other products, and what was your criteria for choosing this product?

A. DEC Object Broker. Scalability was the most important criteria.

Q. What percent of the overall development effort was dedicated to this product?

A. About 60%.

PRODUCT-ORIENTED QUESTIONS

Q. What capabilities in the product did you utilize?

A. Asynchronous processing and integration features with Rogue-Wave and DB Tools.

Q. How did you utilize this product or products?

A. PowerBroker CORBA Plus on both client and server sides.

Q. How easy was it to install the product?

A. Easy enough, so far all ORB products use a command line interface.

Q. How easy was it to develop the project using this product?

A. Easy, especially using integration tools.

Q. What kind of problems did you run into during design and development?

A. Had to build our own trader service, since it was not available.

Q. How easy was it to integrate this product with other products?

A. Easy.

VENDOR-ORIENTED QUESTIONS

Q. How often was the product being updated by the vendor? And did it affect development?

A. Not Applicable

Q. How was customer support/response time during development and maintenance phases?

A. Very responsive, had problems with previous projects utilizing IONA.

Q. What is your overall opinion about the product?

A. Scaleable, very robust.

Q. Would you recommend this product for future use in your organization?

A. Yes.

Visigenic

Response to Customer Questionnaire from Muhammed Rabi, Hughes STX Corp., Tel (301)441-4174.

PROJECT-ORIENTED QUESTIONS

Q. For what project did you use this product?

A. NASA Earth Observatory System (EOS).

Q. What was the nature of the project (communication, finance, etc.)?

A. Implement a proof of concept to replace the current DCE architecture with CORBA-based middleware. Evaluate performance and scalability.

Q. What was the size of the project?

A. Unlimited. Initial phase of 2–3 science analysis servers.

Q. How did the product fit in the project overall architecture?

A. Utilize CORBA as the middleware architecture to connect these analysis servers together.

Q. How long was the project?

A. Started 3 months ago, still in evaluation phase.

Q. What was your development environment (operating system, database, etc.)?

A. NT 4.0, Sun Solaris 5.0, OODB.

Q. What was your runtime environment (if different than development)?

A. Same as development.

Q. Did you consider other products, and what was your criteria for choosing this product?

A. Visigenic VisiBroker for C++, IONA Orbix and Orbixweb, DEC Object Broker. Chose the most popular CORBA products on the market.

Q. What percent of the overall development effort was dedicated to this product?

A. 100%.

PRODUCT-ORIENTED QUESTIONS

Q. What capabilities in the product did you utilize?

A. Naming and event services, waiting on trader service.

Q. How did you utilize this product or products?

A. Will use VisiBroker for C++ on the server side, and IONA OrbixWeb on the client side.

Q. How easy was it to install the product?

A. Easy enough, so far all ORB products use a command line interface. RatIONAl Rose might come up with a mapping to Java.

Q. How easy was it to develop the project using this product?

A. Not Applicable, still in evaluation phase.

Q. What kind of problems did you run into during design and development?

A. Have not yet tested interoperability between VisiBroker for C++ and OrbixWeb.

Q. How easy was it to integrate this product with other products?

A. Server Side—Integrated VisiBroker for C++ with OODB (ObjectStore) Client Side—Used Jbuilder and Symantec Café with OrbixWeb.

VENDOR-ORIENTED QUESTIONS

Q. How often was the product being updated by the vendor? And did it affect development?

A. Not Applicable.

Q. How was customer support/response time during development and maintenance phases?

A. Good for both of Visigenic and IONA. Visigenic has an excellent training program.

Q. What is your overall opinion about the product?

A. Both are good.

Q. Would you recommend this product for future use in your organization?

A. I prefer Visigenic.

Q. Any other comments?

A. IBM is coming up with a new CORBA product in September called IBM Component Broker, which is complete implementation of CORBA 2.0 and all its services. It also includes a framework and an integrated set of tools.

EVALUATION TEST PLAN

This section contains the test plan that was used to demonstrate the client-server communication capabilities of the ORBs. The test is a simple application, which was developed to the CORBA 2.0 standard and uses CORBA 2.0 IDL.

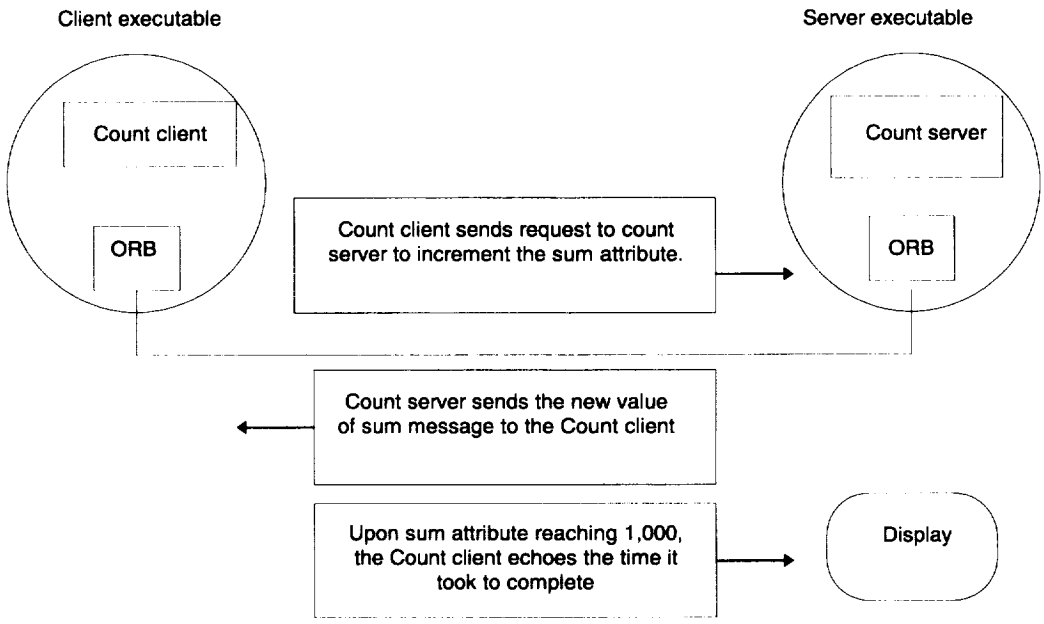
The application, named "Count", consists of a server component and a client component. The server component exports a method called "increment" that simply increments the value of a variable called "sum" and then returns the value to the client. The client program does the following:

- Sets the initial value of the sum attribute.
- Invokes the increment method 1,000 times.

- Displays the final value of sum attribute along with the average ping time it took to increment sum to 1,000.

The evaluation test is depicted in Figure C-1.

Figure C-1. Evaluation Test Diagram



EVALUATION TEST PROCEDURES

This section contains the test procedures for the following products:

- ExperSoft PowerBroker CORBA Plus (C++)
- ExperSoft PowerBroker Java Edition
- IONA Orbix (C++)
- IONA OrbixWeb (Java)
- Visigenic Visibroker for C++
- Visigenic Visibroker for Java.

Each test procedure was run on our development platform. All problems encountered during the installation of the product, building test scripts and running the

test, plus any alterations to the CORBA 2.0 compliant application that were required to execute the application, were documented and considered in the overall evaluation.

ExperSoft PowerBroker CORBA Plus (C++)

SERVER SIDE

Consists of the Server Class that does the following:

1. Initializes the Object Request Broker.
2. Initializes the Basic Object Adapter.
3. Insatiate an instance of the Object Implementation.
4. Creates a Server Instance.
5. Register the Server with the Basic Object Adapter.
6. Prints out a status message.
7. Waits for incoming clients.

Consists of the Count_i Class that does the following:

1. Define the "get" function to return the value of the attribute sum.
2. Define the "set" function to set the value of the attribute sum.
3. Define the "increment" function to increment the value of sum by 1.

CLIENT SIDE

Consists of the Client Class that does the following:

1. Initializes the Object Request Broker.
2. Resolve the URL Object reference.
3. Use reference as if it were a local instance of object implementation.
4. Sets the remote sum attribute to zero.
5. Calculates the start time.
6. Invokes the increment method 1,000 times.

7. Calculates the elapsed time.

8. Prints results.

SCENARIO STEPS

1. Open two MS-DOS windows.

2. Change the directory for both windows to where the test scenarios are located.

   C:\work\exper\cplus

3. Select one window to run the Server using the interpreter.

   > Server.exe

4. The following is displayed in the Server window:

   opening server socket on port: 17337 ready.

5. Go to the other window and run the Client using the interpreter and by adding a new account

   >client 204.255.70.50 17337

6. The following is displayed in the Client window:

   Setting Sum to 0

   Incrementing

   Avg Ping = X where X = the elapsed time in milliseconds

   Sum = 1,000

ExperSoft PowerBroker Java Edition

SERVER SIDE

Consists of the ServerApp Class that does the following:

1. Initializes the Object Request Broker.

2. Initializes the Basic Object Adapter.

3. Insatiate an instance of the Object Implementation.

4. Creates a Server Instance.
5. Register the Server with the Basic Object Adapter.
6. Prints out a status message.
7. Waits for incoming clients.

Consists of the Count_i Class that does the following:

1. Define the "get" function to return the value of the attribute sum.
2. Define the "set" function to set the value of the attribute sum.
3. Define the "increment" function to increment the value of sum by 1.

CLIENT SIDE

Consists of the ClientApp Class that does the following:

1. Initializes the Object Request Broker.
2. Resolve the URL Object reference.
3. Use reference as if it were a local instance of object implementation.
4. Sets the remote sum attribute to zero.
5. Calculates the start time.
6. Invokes the increment method 1,000 times.
7. Calculates the elapsed time.
8. Prints results.

SCENARIO STEPS

1. Open two MS-DOS windows.
2. Change the directory for both windows to where the test scenarios are located.

   C:\work\exper\java

3. Select one window to run the Server using the java interpreter.

   > java ServerApp

4. The following is displayed in the Server window:

   opening server socket on port: 17337 ready.

5. Go to the other window and run the Client using the java interpreter, and by adding a new account.

   > java ClientApp 204.255.70.50 17337

6. The following is displayed in the Client window:

   Setting Sum to 0

   Incrementing

   Avg Ping = X where X = the elapsed time in milliseconds

   Sum = 1,000.

IONA Orbix (C++)

SERVER SIDE

Consists of the Server Class that does the following:

1. Initializes the Object Request Broker.
2. Initializes the Basic Object Adapter.
3. Creates an Server object.
4. Activates the newly created objects.
5. Prints out a status message.
6. Waits for incoming clients.

Consists of the Count_i Class that does the following:

1. Define the "get" function to return the value of the attribute sum.
2. Define the "set" function to set the value of the attribute sum.
3. Define the "increment" function to increment the value of sum by 1.

CLIENT SIDE

Consists of the Client Class that does the following:

1. Initializes the Object Request Broker.
2. Locates a remote ORB.
3. Sets the remote sum attribute to zero.
4. Calculates the start time.
5. Invokes the increment method 1,000 times.
6. Calculates the elapsed time.
7. Prints results.

SCENARIO STEPS

1. From the start menu select Orbix daemon.
2. Open two MS-DOS window.
3. Change directory for both windows to where the test scenarios are located.

   C:\work\orbix\cplus
4. Select one window to run the Server using the Orbix command.

   >putit count C:\work\orbix\cplus\server.exe
5. The following is displayed in the Server window.

   Count Object Created.
6. Go to the other window and run the Client.

   > client 204.255.70.50
7. The following is displayed in the Server window:

   Setting Sum to 0

   Incrementing

   Avg Ping = X where X = the elapsed time in milliseconds.

   Sum = 1,000.

IONA OrbixWeb (Java)

SERVER SIDE

Consists of the Server Class that does the following:

1. Initializes the Object Request Broker.
2. Initializes the Basic Object Adapter.
3. Creates an CountServer object.
4. Activates the newly created objects.
5. Prints out a status message.
6. Waits for incoming clients.

Consists of the CountImp Class that does the following:

1. Defines the "get" to return the value of the attribute sum.
2. Defines the "set" to set the value of the attribute sum.
3. Defines the "increment" to increment the value of sum by 1.

CLIENT SIDE

Consists of the Client Class that does the following:

1. Initializes the Object Request Broker.
2. Locates a remote ORB.
3. Sets the remote sum attribute to zero.
4. Calculates the start time.
5. Invokes the increment method 1,000 times.
6. Calculates the elapsed time.
7. Prints results.

SCENARIO STEPS

1. From the start menu select OrbixWeb daemon.

2. Open two MS-DOS window.
3. Change directory for both windows to where the test scenarios are located.

   C:\work\orbix\java
4. Select one window to run the Server using the java interpreter.

   putit count -java CountServer C:\work\orbix\java
5. Go to the other window and run the Client using the java interpreter, and using the current host name.

   > Java CountClient 204.255.70.50
6. The following is displayed in the Server window:

   Setting Sum to 0

   Incrementing

   Avg Ping = X where X = the elapsed time in milliseconds

   Sum = 1,000.

Visigenic Visibroker for C++

SERVER SIDE

Consists of the Server Class that does the following:

1. Initializes the Object Request Broker.
2. Initializes the Basic Object Adapter.
3. Creates an Server object.
4. Activates the newly created objects.
5. Prints out a status message.
6. Waits for incoming clients.

Consists of the Count_i Class that does the following:

1. Defines the "get" function to return the value of the sum attribute.
2. Defines the "set" function to set the value of the sum attribute.

3. Defines the "increment" to increment the value of sum by 1.

CLIENT SIDE

Consists of the Client Class that does the following:

1. Initializes the Object Request Broker.
2. Locates a remote ORB.
3. Sets the remote sum attribute to zero.
4. Calculates the start time.
5. Invokes the increment method 1,000 times.
6. Calculates the elapsed time.
7. Prints results.

SCENARIO STEPS

1. From the start menu select VisiBroker Smart Agent.
2. Open two MS-DOS window.
3. Change directory for both windows to where the test scenarios are located.

   C:\work\visi\cplus\

4. Select one window to run the Server using the java interpreter.

   > Server.exe

5. The following is displayed in the Server window:

   Count Object Created.

6. Go to the other window and run the Client using the java interpreter, and by adding a new account.

   > Client.exe

7. The following would get displayed in the Server window:

   Setting Sum to 0

   Incrementing

Avg Ping = X where X = the elapsed time in milliseconds

Sum = 1,000.

Visigenic Visibroker for Java

SERVER SIDE

Consists of the Server Class that does the following:

1. Initializes the Object Request Broker.
2. Initializes the Basic Object Adapter.
3. Creates an CountServer object.
4. Activates the newly created objects.
5. Prints out a status message.
6. Waits for incoming clients.

The CountImp Class that does the following:

1. Defines the "get" function to return the value of the attribute sum.
2. Defines the "set" function to set the value of the attribute sum.
3. Defines the "increment" function to increment the value of sum by 1.

CLIENT SIDE

Consists of the Client Class that does the following:

1. Initializes the Object Request Broker.
2. Locates a remote ORB.
3. Sets the remote sum attribute to zero.
4. Calculates the start time.
5. Invokes the increment method 1,000 times.
6. Calculates the elapsed time.
7. Prints results.

SCENARIO STEPS

1. From the start menu select VisiBroker Smart Agent.
2. Open two MS-DOS window.
3. Change directory for both windows to where the test scenarios are located:

   C:\work\visi\java
4. Select one window to run the Server using the java interpreter.

   > java CountServer
5. The following would get displayed in the Server window:

   Count Object Created.
6. Go to the other MS-DOS window and run the Client using the java interpreter.

   > Java CountClient
7. The following would get displayed in the Server window:

   Setting Sum to 0

   Incrementing

   Avg Ping = X where X = the elapsed time in milliseconds

   Sum = 1,000.

EVALUATION CRITERIA MATRIX

The Evaluation criteria matrix presents the results of the Message Broker Product evaluation.

Table C-1. Evaluation Criteria Matrix

Feature	Weight			Score												
	Category	Sub Cat.	Question	0	1	2	3	4	5	ExperSoft		Iona		Visigenic	
PLATFORMS SUPPORTED	**5%**										**19.25**		**22.90**		**22.75**
Server Side		**70%**									**14.00**		**17.50**		**17.50**
· HP-UX 10.20 or above			60%	No		planned	alpha	beta	released	5	10.50	5	10.50	5	10.50
· Windows NT			20%	No		planned	alpha	beta	released	5	3.50	5	3.50	5	3.50
· SGI IRIX v5.3 or above			20%	No		planned	alpha	beta	released	0	0.00	5	3.50	5	3.50
Client Side		**30%**									**5.25**		**5.40**		**5.25**
· AIX			5%	No		planned	alpha	beta	released	5	0.38	5	0.38	5	0.38
· HP-UX 10.20 or above			5%	No		planned	alpha	beta	released	5	0.38	5	0.38	5	0.38
· Java Virtual Machine			40%	No		planned	alpha	beta	released	5	3.00	5	3.00	5	3.00
· Macintosh 7			5%	No		planned	alpha	beta	released	0	0.00	2	0.15	0	0.00
· SGI IRIX v5.3 or above			5%	No		planned	alpha	beta	released	5	0.38	5	0.38	5	0.38
· Sun OS v5.4 or above			5%	No		planned	alpha	beta	released	5	0.38	5	0.38	5	0.38
· Windows 3.1			5%	No		planned	alpha	beta	released	0	0.00	0	0.00	0	0.00
· Windows95			5%	No		planned	alpha	beta	released	5	0.38	5	0.38	5	0.38
· Windows NT			5%	No		planned	alpha	beta	released	5	0.38	5	0.38	5	0.38
LANGUAGE BINDINGS	**5%**										**25.00**		**25.00**		**25.00**
Server Side		**50%**									**12.50**		**12.50**		**12.50**
· C			0%	No	workaround	planned	alpha	beta	released	0	0.00	0	0.00	0	0.00
· C++			65%	No	workaround	planned	alpha	beta	released	5	8.13	5	8.13	5	8.13
· Java			35%	No	workaround	planned	alpha	beta	released	5	4.38	5	4.38	5	4.38
Client Side		**50%**									**12.50**		**12.50**		**12.50**
· C			0%	No	workaround	planned	alpha	beta	released	0	0.00	0	0.00	0	0.00
· C++			35%	No	workaround	planned	alpha	beta	released	5	4.38	5	4.38	5	4.38
· Java			65%	No	workaround	planned	alpha	beta	released	5	8.13	5	8.13	5	8.13
STANDARDS COMPLIANCE	**15%**										**50.93**		**56.55**		**53.55**
CORBA 2.0 Compliance		**50%**									**32.63**		**32.25**		**32.25**
· Does the product support Static Method Invocation?			10%	No		proprietary	CORBA 1.2 compliant		full CORBA 2.0 compliance	5	3.75	5	3.75	5	3.75
· Does the product support Dynamic Method Invocation?			10%	No		proprietary	CORBA 1.2 compliant		full CORBA 2.0 compliance	5	3.75	5	3.75	5	3.75
· Does the product support CORBA Interface Repository?			5%	No		proprietary	CORBA 1.2 compliant		full CORBA 2.0 compliance	5	1.88	5	1.88	5	1.88
· Does the product support Server Callbacks?			5%	No	planning for full CORBA 2.0 compliance	proprietary	partial CORBA 2.0 compliance		full CORBA 2.0 compliance	2	0.75	1	0.38	1	0.38
· What type of IDL to C++ mapping does the product support?			10%	No	planning for full CORBA 2.0 compliance	proprietary	partial CORBA 2.0 compliance		full CORBA 2.0 compliance	5	3.75	5	3.75	5	3.75
· What type of IDL to java mapping does the product support?			10%	No	planning for full CORBA 2.0 compliance	proprietary	sun idl2java mapping		full CORBA 2.0 compliance	2	1.50	2	1.50	3	2.25
· What type of API does the product support?			5%	No		proprietary	CORBA 1.2 compliant		full CORBA 2.0 compliance	5	1.88	5	1.88	5	1.88
· Does the product support the Internet Inter-ORB Protocol (IIOP)?			30%	No					yes	5	11.25	5	11.25	5	11.25

Feature	Weight			Score													
	Cate-gory	Sub Cat.	Question	0	1		3	4	5	ExperSoft		Iona		Visigenic	
Will the product supp ortable Object Adapter (POA)?			5%	No		after Q2 1998	before or at Q2 1998	Q4 1997	Q3 1997	3	1.13	3	1.13	3	1.13
Will the product support Interoperable Object Reference (IOR)?			5%	No		after Q2 1998	before or at Q2 1998	Q4 1997	Q3 1997	3	1.13	3	1.13	3	1.13
Does the product supply an OLE 2.0 Bridge?			5%	No			partial		Yes	5	1.88	5	1.88	3	1.13
CORBA Services		**50%**									**18.30**		**24.30**		**21.30**
Life Cycle Service															
Does the product provide operations for creating, copying and deleting objects?			10%	no	workaround	integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 life cycle service compliant	3	2.25	1	0.75	3	2.25
Persistent Service															
Does the product provide an interface for storing objects on a variety of storage servers?			10%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 persistent service compliant	3	2.25	5	3.75	0	0.00
Naming Service															
Does the product allow objects on the bus to locate other objects by name?			15%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 naming service compliant	5	5.63	5	5.63	5	5.63
Event Service															
Does the product allow objects to register and unregistered dynamically for events?			14%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 event service compliant	5	5.25	5	5.25	5	5.25
Concurrency Control Service															
Does the product provide concurrency control service?			8%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 concurrency service compliant	3	1.80	3	1.80	3	1.80
Transaction Processing Service															
Does the product support transaction services?			5%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 transaction service compliant	3	1.13	5	1.88	3	1.13
Relationship Service															
Does the product provide operations to dynamically create associations between objects that know nothing of each other?			8%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products	CORBA 2.0 relationship service compliant	0	0.00	0	0.00	0	0.00
Externalization Service															
Does the product provide a standard way of getting data into and out of an object?			8%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products	CORBA 2.0 externalization service compliant	0	0.00	0	0.00	0	0.00

Feature	Weight			Score													
	Cate-gory	Sub Cat.	Question	0	1		3	4	5	ExperSoft		Iona		Visigenic	
Query Service															
· Does the product provide query operations for objects?			8%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 query service compliant	0	0.00	0	0.00	0	0.00
Security Service															
· Does the product security service provide identification and authentication?			5%	no			proprietary	planning for full CORBA 2.0 compliance	CORBA 2.0 security service compliant	0	0.00	5	1.88	5	1.88
· Does the product security service support privilege delegation?			4%	no			proprietary	planning for full CORBA 2.0 compliance	CORBA 2.0 security service compliant	0	0.00	5	1.50	5	1.50
· Does the product security service support transaction authorization?			3%	no			proprietary	planning for full CORBA 2.0 compliance	CORBA 2.0 security service compliant	0	0.00	5	1.13	5	1.13
· If product does not support security service, does it integrate with other Security Package - such as a DCE based one?			2%	no					built-in support	0	0.00	5	0.75	5	0.75
Licensing Service															
· Does the product provide operations for metering the use of objects?			0%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 licensing service compliant	0	0.00	0	0.00	0	0.00
Properties Service															
· Does the product provide operations to dynamically associate properties with object state?			0%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 properties service compliant	3	0.00	0	0.00	0	0.00
Timing Service															
· Does the product provide operations for defining and managing time triggered-events?			0%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 timing service compliant	0	0.00	0	0.00	0	0.00
Trader Service															
· Does the product provide means for objects to publicize their services and bid for jobs?			0%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 trader service compliant	0	0.00	5	0.00	2	0.00
Collection Service															
· Does the product provide interfaces to generically create and manipulate the most common collections.?			0%	no		integrates with other products services	proprietary	integrates with other CORBA 2.0 products services	CORBA 2.0 collection service compliant	0	0.00	0	0.00	0	0.00
USABILITY AND CUSTOMIZABILITY	**15%**										**66.53**		**55.13**		**64.80**
Ease of Installation		**10%**									**5.85**		**7.20**		**4.80**

Feature	Weight			Score													
	Cate-gory	Sub Cat.	Question	0	1		3	4	5	ExperSoft		Iona		Visigenic	
· Does the product pro... automated installation?			50%	no			partial		yes	3	2.25	5	3.75	3	2.25
· Does the product installs without damaging the current environment?			40%	no			partial		yes	5	3.00	5	3.00	3	1.80
· How do you rate installation time with similarly complex and sized products on that platform?			10%	no	slow	slower than average	average	faster than average	fast	4	0.60	3	0.45	5	0.75
Ease of configuration		10%									7.50		7.50		7.50
· Does the product provide some capability that allows additions or modifications to the infrastructure?			100%	no			partial		yes	5	7.50	5	7.50	5	7.50
Portability (Same Vendor)		40%									30.00		18.00		30.00
· Is the server-side code easily portable across platforms?			100%	no		with considerable changes	with minimal changes to code	with minimal changes to env	no change required	5	30.00	3	18.00	5	30.00
Portability (Different Vendor)		20%									10.50		10.50		10.50
· Is the server-side code easily portable to different ORB products?			50%	no		with considerable changes	with minimal changes to code	with minimal changes to env	no change required	3	4.50	3	4.50	3	4.50
· Does the ORB have demonstrated interoperability with any other commercial ORB?			50%	none		only with one product	with a few other products	with most products	with all other products	4	6.00	4	6.00	4	6.00
Documentation		10%									5.18		5.93		6.00
· Is the documentation accurate and detailed?			25%	no			partial		completely accurate	3	1.13	5	1.88	5	1.88
· Are the documentation examples effective in illustrating important usage points?			25%	no			partial		yes	3	1.13	3	1.13	4	1.50
· Does the documentation provide adequate examples for all platforms supported?			25%	no			uses one platform and assumes the rest will follow	separate examples for unix vs NT	separate examples for each platform	4	1.50	4	1.50	4	1.50
· What are the available formats for documentation?			10%	hardcopy			online		hard copy and online	5	0.75	5	0.75	3	0.45
· Does the product provide prepackaged code or libraries?			15%	none			uses libs provided by other vendors		provides a complete set of libraries	3	0.68	3	0.68	3	0.68
Performance		10%									7.50		6.00		6.00
· Using the custom test script what is the average time for one client to call a server?			25%	>=100 msec	75 -99 msec	50 -74 msec	10-49 msec	5-9 msec	< 5 msec	5	1.88	4	1.50	4	1.50
· Using the custom test script what is the average time for 10 clients to call one server?			75%	>=1000 msec	750 -999 msec	500 -749 msec	100-499 msec	50-99 msec	< 5 msec	5	5.63	4	4.50	4	4.50
FUNCTIONAL FEATURES	20%										71.00		87.00		83.00
Load Balancing		10%									0.00		6.00		6.00
· Does the product support load balancing			100%	no			through proprietary trans monitor		full blown J16Odynamic load balancing	0	0.00	3	6.00	3	6.00
Audit Trail		10%									0.00		10.00		10.00
· Does the product provide audit capability?			100%	no			limited		yes	0	0.00	5	10.00	5	10.00
· Does the product provide interface to audit operating system activities?			0%	no			limited		yes	0	0.00	0	0.00	0	0.00
· Does the product provide interface to Sybase audit capabilities?			0%	no			limited		yes	0	0.00	0	0.00	0	0.00
Error Management		20%									13.00		13.00		13.00
· Does the product provide error handling capability?			65%	no			limited		yes	5	13.00	5	13.00	5	13.00
· Does the product provide interface to operating system errors?			35%	no			limited		yes	0	0.00	0	0.00	0	0.00

Feature	Weight			Score													
	Cate-gory	Sub Cat.	Question	0	1		3	4	5	ExperSoft		Iona		Visigenic	
· Does the product pr .terface to Sybase errors?			0%	no			limited		yes	0	0.00	0	0.00	0	0.00
Interoperability		40%									40.00		40.00		40.00
· Does the product provide cross platforms independence?			45%	no			with minimal changes		no change required	5	18.00	5	18.00	5	18.00
· Does the product provide cross language independence?			45%	no			with minimal changes		no change required	5	18.00	5	18.00	5	18.00
· Can the product be used with an OLE client?			10%	no			through an API		built in support for OLE	5	4.00	5	4.00	5	4.00
Fault-resilience		20%									18.00		18.00		14.00
· Does the product support fail-over (standby servers) for fault recovery?			25%	no			partial		yes	3	3.00	3	3.00	5	5.00
· Does the product support persistent queuing (allowing asynchronous, guaranteed messaging)?			75%	no			defered synch		asynch	5	15.00	5	15.00	3	9.00
DEVELOPMENT ISSUES	15%										61.35		56.55		58.95
Development Environment		40%									27.60		22.80		25.20
· How easy is development effort using this product?			40%		v. difficult	more difficult than average	average	easier than average	v. easy	4	9.60	4	9.60	5	12.00
· Is distributed object development using this product easily integrated with an open, integrated development environment?			20%	no			with some modification		built-in support	5	6.00	3	3.60	3	3.60
· Do any language development environments generate IDL compatible with this product?			20%	no					yes	5	6.00	5	6.00	5	6.00
· Does the product integrate with an analysis/design tool?			20%	no			works with Rational Rose and RogueWave		built-in integration with Rational Rose and Rogue Wave	5	6.00	3	3.60	3	3.60
Debugging Aids		60%									33.75		33.75		33.75
· Does the product provide method tracing at any granularity?			25%	no					yes	5	11.25	5	11.25	5	11.25
· Can the product log method / messages to a log file?			25%	no			partial log		yes	5	11.25	5	11.25	5	11.25
· Can the product trace daemon activation?			25%	no					yes	0	0.00	0	0.00	0	0.00
· Can the product trace the instantiation and destruction of objects in the server?			25%	no					yes	5	11.25	5	11.25	5	11.25
BUSINESS ISSUES	10%										40.40		37.10		49.25
Technical Support		40%									14.40		12.60		20.00
· What level of technical support does the vendor provide?			30%	no support	BBS	plus fax	plus email	plus www	plus hot-line phone support	3	3.60	3	3.60	5	6.00
· Does the vendor provide on-site consulting?			15%	no					yes	5	3.00	5	3.00	5	3.00
· Does the vendor provide extended service agreement?			10%	no					yes	0	0.00	0	0.00	5	2.00
· What is the turn-around for vendor response?			15%	> 96 hrs	> 72 hrs and <=96 hrs	> 48 hrs and <=72 hrs	> 24hrs and <= 48 hrs	> 4 hrs and <= 24 hrs	< 4 hrs	3	1.80	2	1.20	5	3.00
· Are support responses to customer questions accurate?			15%	no	very rarely		most of the time		always	5	3.00	5	3.00	5	3.00
· Does the vendor response to customer problems effectively (urgent patches)?			15%	no					yes	5	3.00	3	1.80	5	3.00
Viability of the vendor		25%									12.50		10.00		12.50
· Is the vendor well capitalized?			50%	no					yes	5	6.25	5	6.25	5	6.25
· Does the vendor have sufficient technical staff to evolve and support the product?			50%	no	not enough		average		more than enough	5	6.25	3	3.75	5	6.25
Dedication of the vendor to the product		25%									10.50		10.50		11.75

Feature	Weight			Score													
	Cate-gory	Sub Cat.	Question	0	1		3	4	5	ExperSoft		Iona		Visigenic	
· Does the vendor has an influential representation at OMG?			30%	no		OMG member	involved with one RFP	involved with more than one RFP	plus std commity member	5	3.75	5	3.75	4	3.00
· Can the vendor respond effectively to changing needs of the industry or of its key customers?			40%	no	rarely		some of the time		always	3	3.00	3	3.00	5	5.00
· Is the product part of the vendor's long-term, strategic direction?			30%	no					yes	5	3.75	5	3.75	5	3.75
Strategic alliance partners		**10%**									**3.00**		**4.00**		**5.00**
· Has the vendor formed alliances with significant third-party providers?			100%	none			few industry leaders		most industry leaders	3	3.00	4	4.00	5	5.00
MARKET ACCEPTANCE	**10%**										**33.25**		**43.25**		**43.25**
Maturity		**35%**									**10.50**		**14.00**		**14.00**
· When was the product introduced?			50%	just introduced	within the last month	more than a month and less than a year	more than one year and less than two years	within two to three years	three years or more	5	8.75	5	8.75	5	8.75
· How often was the product updated?			50%	more than four times per year	four times per year	three times per year	twice per year	once per year	never been updated since	1	1.75	3	5.25	3	5.25
Market Share		**65%**									**22.75**		**29.25**		**29.25**
· What is the current market share relative to competitors?			50%	0%-3%	3%-5%	5%-10%	11%-30%	31%-60%	> 60%	2	6.50	4	13.00	4	13.00
· What is the size of the current installed base?			25%	0-25	26-50	51-100	101-500	501-1000	>1000	5	8.13	5	8.13	5	8.13
· What is the largest current installation of the product?			25%	0-10	11-25	26-50	51-100	101-200	>200	5	8.13	5	8.13	5	8.13
COST & TRAINING	**5%**										**11.25**		**13.75**		**11.25**
· For a 5 servers and 40 clients ASAC EA system, what is the product suite price range?			50%	> 30,000	25,001-30,000	20,001-25,000	15,001-20,000	10,001-15,000	<= 10,000	3	7.50	4	10.00	3	7.50
· What is the cost of one week of consulting/training per student?			25%	> 3,000	2,501-3,000	2,001-2,500	1,501-2,000	1,001-1,500	<=1,000	3	3.75	3	3.75	3	3.75
· What is the cost of upgrades relative to current version cost?			25%	>75%	51%-75%	26% - 50%	11% - 25%	6% -10%	<= 5%	0	0.00	0	0.00	5	6.25
TOTAL SCORE											**378.95**		**397.23**		**411.80**

ATTACHMENT A – CORBA ORB VENDOR QUESTIONNAIRE RESPONSES

The following questions were asked to potential vendors as part of the CORBA ORB product selection process. The responses from ExperSoft, IONA, and Visigenic are contained in this attachment, following the list of questions. Note: we received detailed responses from BEA Systems, Inc., but they are not included in this attachment since BEA was not a final candidate.

Contents

QUESTIONS TO VENDOR

- PLATFORMS SUPPORTED
  - Server Side
    - HP-UX 10.20 or above
    - Windows NT
    - SGI IRIX v5.3 or above
  - Client Side
    - AIX
    - HP-UX 10.20 or above
    - Java Virtual Machine
    - Macintosh 7
    - SGI IRIX v5.3 or above
    - Sun OS v5.4 or above
    - Windows 3.1
    - Windows95
    - Windows NT
- LANGUAGE BINDINGS
  - Server Side
    - C
    - C++
    - Java

- Client Side
  - C
  - C++
  - Java

- STANDARDS COMPLIANCE
  - CORBA 2.0 Compliance
    - Does the product support Static Method Invocation?
    - Does the product support Dynamic Method Invocation?
    - Does the product support CORBA Interface Repository?
    - Does the product support Server Callbacks?
    - What type of IDL to C++ mapping does the product support?
    - What type of IDL to Java mapping does the product support?
    - What type of API does the product support?
    - Does the product support the Internet Inter-ORB Protocol (IIOP)?
    - Will the product support Portable Object Adapter (POA)?
    - Will the product support Interoperable Object Reference (IOR)?
    - Does the product supply an OLE 2.0 bridge?
  - CORBA Services
    - Life Cycle Service
      - ⇒ Does the product provide operations for creating, copying, and deleting objects?
    - Persistent Service
      - ⇒ Does the product provide an interface for storing objects on a variety of storage servers?
    - Naming Service

⇒ Does the product allow objects on the bus to locate other objects by name?

- Event Service

  ⇒ Does the product allow objects to register and unregister dynamically for events?

- Concurrency Control Service

  ⇒ Does the product provide concurrency control service?

- Transaction Processing Service

  ⇒ Does the product support transaction services?

- Relationship Service

  ⇒ Does the product provide operations to dynamically create associations between objects that know nothing of each other?

- Externalization Service

  ⇒ Does the product provide a standard way of getting data into and out of an object?

- Query Service

  ⇒ Does the product provide query operations for objects?

- Security Service

  ⇒ Does the product security service provide identification and authentication?

  ⇒ Does the product security service support privilege delegation?

  ⇒ Does the product security service support transaction authorization?

  ⇒ If product does not support security service, does it integrate with another Security Package, such as a DCE-based one?

- Licensing Service

  ⇒ Does the product provide operations for metering the use of objects?

- Properties Service

  ⇒ Does the product provide operations to dynamically associate properties with object state?

- Timing Service

  ⇒ Does the product provide operations for defining and managing time-triggered events?

- Trader Service

  ⇒ Does the product provide means for objects to publicize their services and bid for jobs?

- Collection Service

  ⇒ Does the product provide interfaces to generically create and manipulate the most common collections?

◆ USABILITY AND CUSTOMIZABILITY

➤ Ease of Installation

- Does the product provide automated installation?
- Does the product install without damaging the current environment?
- How do you rate installation time with similarly complex and sized products on that platform?

➤ Ease of Configuration

- Does the product provide some capability that allows additions or modifications to the infrastructure?

➤ Portability (Same Vendor)

- Is the server-side code easily portable across platforms?

➤ Portability (Different Vendor)

- Is the server-side code easily portable to different ORB products?
- Does the ORB have demonstrated interoperability with any other commercial ORB?

- Documentation
  - Is the documentation accurate and detailed?
  - Are the documentation examples effective in illustrating important usage points?
  - Does the documentation provide adequate examples for all platforms supported?
  - What are the available formats for documentation?
  - Does the product provide prepackaged code or libraries?
- Performance
  - Using the custom test script, what is the average time for one client to call a server?
  - Using the custom test script, what is the average time for 10 clients to call one server?

◆ FUNCTIONAL FEATURES

- Load Balancing
  - Does the product support load balancing?
- Audit Trail
  - Does the product provide audit capability?
  - Does the product provide interface to audit operating system activities?
  - Does the product provide interface to Sybase audit capabilities?
- Error Management
  - Does the product provide error handling capability?
  - Does the product provide interface to operating system errors?
  - Does the product provide interface to Sybase errors?
- Interoperability
  - Does the product provide cross-platforms independence?
  - Does the product provide cross language independence?

- Can the product be used with an OLE client?

- Fault-resilience

  - Does the product support fail-over (standby servers) for fault recovery?

  - Does the product support persistent queuing (allowing asynchronous, guaranteed messaging)?

- DEVELOPMENT ISSUES

  - Development Environment

    - How easy is development effort using this product?

    - Is distributed object development using this product easily integrated with an open, integrated development environment?

    - Do any language development environments generate IDL compatible with this product?

    - Does the product integrate with an analysis/design tool?

  - Debugging Aids

    - Does the product provide method tracing at any granularity?

    - Can the product log method/messages to a log file?

    - Can the product trace daemon activation?

    - Can the product trace the instantiation and destruction of objects in the server?

- BUSINESS ISSUES

  - Technical Support

    - What level of technical support does the vendor provide?

    - Does the vendor provide on-site consulting?

    - Does the vendor provide extended service agreements?

    - What is the turn-around for vendor response?

    - Are support responses to customer questions accurate?

- Does the vendor response to customer problems effectively (urgent patches)?

➢ Viability of the Vendor

- Is the vendor well-capitalized?
- Does the vendor have sufficient technical staff to evolve and support the product?

➢ Dedication of the Vendor to the Product

- Does the vendor have an influential representation at OMG?
- Can the vendor respond effectively to changing needs of the industry or of its key customers?
- Is the product part of the vendor's long-term, strategic direction?

➢ Strategic Alliance Partners

- Has the vendor formed alliances with significant third-party providers?

◆ MARKET ACCEPTANCE

➢ Maturity

- When was the product introduced?
- How often was the product updated?

➢ Market Share

- What is the current market share relative to competitors?
- What is the size of the current installed base?
- What is the largest current installation of the product?

◆ COST & TRAINING

➢ For a 5-server and 40-client ASAC EA system, what is the product suite price range?

➢ What is the cost of one week of consulting/training per student?

➢ What is the cost of upgrades relative to current version cost?

RESPONSE FROM EXPERSOFT CORPORATION

PLATFORMS SUPPORTED	
AIX	Yes
HP-UX v9.0 or above	Yes
Java Virtual Machine	Yes
Macintosh 7	
SGI IRIX v5.3 or above	No
Sun OS v5.4 or above	Yes
Windows 3.1	
Windows95	Yes
Windows NT	Yes
LANGUAGE BINDINGS	
C	
C++	Yes
COBOL	
Java	Yes
Java Beans	
PERL	
STANDARDS COMPLIANCE	
CORBA 2.0 Compliance	
Does the product support the full CORBA 2 IDL?	Yes
Does the product support the full CORBA 2 API?	Yes
Does the product support the Internet Inter-ORB Protocol (IIOP)?	Yes
Does the product support TCP/IP?	Yes

Scalability

Can the current product be scaled-up to 10 servers?	Yes
Can the current product be scaled-up to 25 servers?	Yes
Can the current product be scaled-up to 100 servers?	Yes
Will scalability effect performance?	Minimally
Does the product support replication of servers?	Yes

Interoperability

Does the product provide cross platforms independence?	Yes
Does the product provide cross language independence?	Yes
Can the product be used with an OLE client?	Yes
Is there a limit on the number of clients that a server ORB product can support?	Theoretical no
How many clients can a server product support?	Platform dictates

Fault-resilience

Does the product support fail-over (standby servers) for fault recovery?	Yes
Does the product support persistent queuing	Yes
(allowing asynchronous, guaranteed messaging)?	Yes

USABILITY AND CUSTOMIZABILITY

Ease of Installation

Does the product utilize an effective script for its installation?	Yes
Does the product installs without damaging the current environment?	Yes
Is the amount of time for installation consistent with similarly complex and sized products on the platform?	No, less time

Ease of configuration

Does the product provide some capability that allows additions or modifications to the infrastructure?	Customizations are possible

Portability of Implementation (Same Vendor)	
Is the server-side code easily portable across platforms?	Yes.
Portability of Implementation (Different Vendor)	
Is the server-side code easily portable to different ORB product implementations?	It's been done
Does the ORB have demonstrated interoperability with any other commercial ORB?	Yes
Documentation	
Is the documentation correct?	Yes
Are the documentation examples effective in illustrating important usage points?	Yes
Does the documentation provide adequate examples for all platforms supported?	Yes
What are the available formats for documentation?	Paper/HTML
What is the availability of prepackaged code or libraries?	Today
FUNCTIONAL FEATURES	
Concurrency	
Does the product support multithreading	Yes
Is the thread package platform independent, e.g., POSIX compliant?	Yes
Does the product support distributed threads?	No
Database Interfaces	
Does the product provide interfaces to SQL?	No
Does the product provide interfaces to Sybase?	No
Remote Access	
Does the product provide remote app startup and shutdown?	BOA activation?
Does the product provide scripting capabilities?	No

State Maintenance	
Can the product save object state?	No
Software Distribution	
Does the product support push technology?	Yes
Load Balancing	
Does the product support load balancing	Yes
Audit Trail	
Does the product provide audit capability?	No
Does the product provide interface to audit operating system activities?	No
Does the product provide interface to Sybase audit capabilities?	No
Error Management	
Does the product provide error handling capability?	Yes
Does the product provide interface to operating system errors?	No
Does the product provide interface to Sybase errors?	No
CORBA/Non-CORBA SERVICES	
Life Cycle Service	
Does the product provide operations for creating, copying and deleting objects?	Yes
Is the product life cycle service CORBA 2.0 compliant?	Yes
Persistent Service	
Does the product provide an interface for storing objects on a variety of storage servers?	Yes
Is the product persistent service CORBA 2.0 compliant?	Partially
Naming Service	
Does the product allow objects on the bus to locate other objects by name?	Yes

Is the product naming service CORBA 2.0 compliant?	Yes
Event Service	
Does the product allow objects to register and unregister dynamically for events?	Yes
Is the product event service CORBA 2.0 compliant?	Yes
Concurrency Control Service	
Does the product provide concurrency control service?	No
Is the product concurrency control service CORBA 2.0 compliant?	No
Transaction Processing Service	
Does the product support some means of transaction management?	Yes
Is the product transaction service CORBA 2.0 compliant?	Yes
If product does not support transaction service, does it integrate with a transaction processing product?	Yes also
Relationship Service	
Does the product provide operations to dynamically create associations between objects that know nothing of each other?	Yes
Is the product relationship service CORBA 2.0 compliant?	Yes
Externalization Service	
Does the product provide a standard way of getting data into and out of an object?	This year
Is the product externalization service CORBA 2.0 compliant?	This year
Query Service	
Does the product provide query operations for objects?	No
Is the product query service CORBA 2.0 compliant?	No
Security Service	
Does the product provide security service?	SSL
Is the product security service CORBA 2.0 compliant?	No

Does the product security service provide identification and authentication?	No
Does the product security service support privilege delegation?	No
Does the product security service support transaction authorization?	No
If product does not support security service, does it integrate with a other Security Package - such as a DCE based one?	No
Can the product integrate with a system that provides a security component?	Yes
Licensing Service	
Does the product provide operations for metering the use of objects?	No
Is the product licensing service CORBA 2.0 compliant?	N/A
Properties Service	
Does the product provide operations to dynamically associate properties with object state?	This year
Is the product properties service CORBA 2.0 compliant?	This year
Timing Service	
Does the product provide operations for defining and managing time triggered-events?	No
Is the product timing service CORBA 2.0 compliant?	No
Trader Service	
Does the product provide means for objects to publicize their services and bid for jobs?	This year
Is the product trader service CORBA 2.0 compliant?	This year
Collection Service	
Does the product provide interfaces to generically create and manipulate the most common collections.?	Yes
Is the product collection service CORBA 2.0 compliant?	No

DEVELOPMENT ISSUES

Development environment

How easy is development effort using this product?	Very easy
Is distributed object development using this product easily integrated with an open, integrated development environment?	Yes
Do any language development environments generate IDL compatible with this product?	Yes
Does the product integrate with an analysis/design tool?	Ptech & Rose

Debugging Aids

Does the product provide method tracing at any granularity?	Yes
Can the product log method / messages to a log file?	Yes
Can the product trace daemon activation?	Yes
Can the product trace the instantiation and destruction of objects in the server?	Yes

BUSINESS ISSUES

Technical Support

Does the vendor provide a dedicated hot-line phone service?	Yes
Does the vendor provide electronic mail and fax service of support request?	Yes
Does the vendor provide World Wide Web support?	Yes
Does the vendor provide on-site support advise/troubleshooting?	Yes
Does the vendor provide extended service agreement?	Yes
Is the vendor responsive?	Always
Are support responses to customer questions accurate?	Yes
Is consulting help on the product available?	Yes
Does the vendor response to customer problems effectively (urgent patches)?	Yes

Viability of the vendor	
Is the vendor well capitalized?	Yes
Does the vendor have seasoned management staff?	Yes
Does the vendor have sufficient technical staff to evolve and support the product?	Yes
Can the vendor respond effectively to changing needs of the industry or of its key customers?	Yes
Dedication of the vendor to the product	
Is the product part of the vendor's long-term, strategic direction?	Yes
Strategic alliance partners	
Has the vendor formed alliances with significant third-party providers?	Yes
MARKET ACCEPTANCE	
Maturity	
When was the product introduced?	1994
How often was the product updated?	4 time a year
Market Share	
What is the current market share?	Under NDA only
What is the size of the current installed base?	Under NDA only
What is the largest current installation of the product?	Under NDA only
What is the current market share relative to competitors?	Under NDA only
TRAINING	
Does vendor provide development training?	Yes
Does vendor provide maintenance training?	Yes
Does vendor provide administrative training?	Yes
Can unscheduled training be arranged?	Yes

COST

Is product currently on GSA schedule?	WWW/Sales Dept.
If not, does vendor provide not-for-profit pricing?	WWW/Sales Dept.
Is the product suite competitively priced to deploy for ASAC?	WWW/Sales Dept.
For a 5 servers and 40 clients ASAC AE system, what is the product suite price range?	WWW/Sales Dept.
Is the extended service support included in price?	WWW/Sales Dept.
If not, what is the extended service support cost?	WWW/Sales Dept.
Is training included priced?	WWW/Sales Dept.
If not, what is training cost?	WWW/Sales Dept.
What is the price for technical support?	WWW/Sales Dept.
What is the cost of upgrades relative to current version cost?	WWW/Sales Dept.

RESPONSE FROM IONA CORPORATION

-PLATFORMS SUPPORTED

al Machine *Yes.*

Macintosh 7 *Yes (but only with Orbix 1.3.5)*

SGI IRIX v5.3 or above *Yes.*

Sun OS v5.4 or above *Yes.*

Windows 3.1 *No.*

Windows95 *Yes.*

Windows NT *Yes.*

LANGUAGE BINDINGS

C No (but it is possible to wrap legacy C applcations in C++)

C++ Yes.

COBOL Yes.

Perl No.

Java Yes.

Java Beans Will support in Q3/Q4 timeframe.

STANDARDS COMPLIANCE

CORBA 2.0 Compliance

Does the product suppor**STANDARDS COMPLIANCE**

CORBA 2.0 Compliance

ct support the full CORBA 2 API? Yes.

Does the product support the Internet Inter-ORB Protocol (IIOP)? Yes.

Does the product support TCP/IP? Yes.

Scalability

Can the current product be scaled-up to 10 servers? **Scalability**

urrent product be scaled-up to 25 servers? Yes.

Can the current product be scaled-up to 100 servers? Yes. There are Orbix applications in production with several hundred servers and several thousand clients.

Will scalability effect performance? No. Orbix was designed to be both scaleable and efficient. Adding new servers to the system has no effect on the performance of existing servers.

Does the product support replication of servers? Yes. Replication of servers is supported through the various activation modes available. However, Orbix does not maintain state across replicated servers.

Interoperability

Does the product provide cross platforms indep**Interoperability**

he product provide cross language independence? Yes.

Can the product be used with an OLE client? Yes. OLE servers are also supported.

Is there a limit on the number of clients that a server ORB product can support? There is no Orbix imposed limit. Deployed Orbix applications exist with 5,000 clients.

How many clients can a server product support? Again, there is no Orbix imposed limit. It then becomes a question of allocating system resources. We have demonstrated 2,000 clients simultaneously connected to a NT server.

Fault-resilience

Does the product support fail-over (standby serve**Fault-resilience**

very?

Orbix is a tool for building scaleable distributed systems. It doesOrbix is a tool for building scaleable distributed systems. It does not provide a shrink wrapped fault tolerant mechanism, instead it offers access to internal 'hooks' for incorporating fault tolerance functionality. This is in keeping with our approach to multi-threaded servers; we expose the threading mechanism for maximum flexibility but also provide standard threading filters to allow for common threading policies. In this respect we have a number of fault tolerant options, mostly using a feature called a 'Smart Proxy', but we can also provide a standard smart proxy to handle fault tolerance.

Enhanced fault tolerance capabilities figure highly in IONA's future direction with both the Orbix Transaction Monitor (OTM) and the internal structure of Orbix. Orbix OTM is

IONA's product offering in the TM space, which seeks to offer a coherent set of high-value components on top of a proven, reliable communications infrastructure. Orbix OTM, will support resilience of stateless servers in two ways: firstly, via activation modes, and secondly via transparent re-establishment of failed connections.

OrbixTalk (IONA's asynchronous messaging solution) also allows for communication between applications that may or may not active at the same time. Using a store and forward mechanism in conjunction with a persistent store this allows for the guaranteed delivery of messages to clients and also ensures they arrive in chronological order.

- Does the product support persistent queuing (allowing asynchronous, guaranteed messaging)?

Yes. OrbixTalk supports persistent queuing via a store and forward mechanism and a persistent store. OrbixTalk is based on IP Multicast technology making it both scaleable and performant.

USABILITY AND CUSTOMIZABILITY

Ease of Installation

- Does the product utilize an effective script for its installation? *Yes. The install scripts used are the standard ones for those platforms. For instance, Orbix for NT/95 has a Microsoft certified install script, as does OrbixWeb. On Solaris, pkgadd is used, and on HP-UX 10.20 swinstall is used.*

- Does the product installs without damaging the current environment? *Yes.*

- Is the amount of time for installation consistent with similarly complex and sized products on the platform? *Yes.*

Ease of configuration

- Does the product provide some capability that allows additions or modifications to the infrastructure?

Portability of Implementation (Same Vendor)

- Is the server-side code easily portable across platforms? *Yes. Because all versions of Orbix conform to the CORBA 2.0 standard the code will be similar across platforms. The only differences result from differences in compilers. For example, if a compiler does not support native exception handling then the code will be slightly different than for a compiler that does.*

Portability of Implementation (Different Vendor)

- Is the server-side code easily portable to different ORB product implementations? *Currently, there are differences between the server side code that needs to be written for different ORB vendors. For some ORBs, the different will be minimal, while for others there may be a significant difference.*

- Does the ORB have demonstrated interoperability with any other commercial ORB? *Yes, we test in-house against five or six ORBs including NEO, Visigenics, HP ORB Plus and HP's Distributed SmallTalk. There is a web-site (http://corbanet.dstc.edu.au/) that demonstrated interoperability between all major CORBA compliant ORBs.*

Documentation

- Is the documentation correct? *Yes. Orbix has always been praised for the high quality of its documentation. Orbix comes with both a Programming Guide and a comprehensive Reference Guide.*

- Are the documentation examples effective in illustrating important usage points? *Yes, examples are given to illustrate all the topics discussed in the manuals.*

- Does the documentation provide adequate examples for all platforms supported? *Because the use of Orbix is basically identical across UNIX platforms these are treated as one, and a distinction is made between these and the use of Orbix on NT.*

- What are the available formats for documentation? *The documentation is available in both hard copy and electronic formats.*

- What is the availability of prepackaged code or libraries? *I am not quite sure what this refers to. Orbix comes with extensive fully coded examples.*

FUNCTIONAL FEATURES

Concurrency

- Does the product support multithreading *Yes, Orbix fully supports multi-threading, and the Orbix libraries are thread-safe. However, it is still the responsibility of the application programmer to handle mutex's etc. Threading is supported by Thread Filters. A thread filter implements a particular threading policy. Orbix comes prebuilt thread filters supporting the following threading policies:*

  thread per request, pool of threads, thread per object, thread per client. These filters can of course be customised to implement alternative policies.

- Is the thread package platform independent, e.g., POSIX compliant? *By default Orbix makes use of native threading packages, which often are POSIX compliant. Orbix also supports the ability to switch in a third-party threading package such as threads.h++ from RogueWave. These packages may or may not be platform independent.*

- Does the product support distributed threads? *We at IONA have not heard of distributed threads.*

Database Interfaces

- Does the product provide interfaces to SQL? *There are two approaches that can be taken to integrate an ORB with a database. The simplest is "front-ending" where the database is wrapped in IDL. A client makes a call on the IDL interface and within the implementation code SQL calls are made on the database. This type of integration requires no specific integration between the ORB and the database and of course is possible with Orbix.*

  The second approach is to make Orbix objects transparently persistent in the database, allowing the objects to live longer than the process in which they are contained. This approach requires a specific integration with the ORB. Currently, there are a few third-party companies with products that provide this support and handle the difficult issue of object-to-relatIONAl mapping. They are Persistence Inc, ONTOS and UniSQL.

- Does the product provide interfaces to Sybase? *Yes, the products from the companies listed above provide an integration between Orbix and Sybase.*

Remote Access

- Does the product provide remote app startup and shutdown *The Orbix Manager provides the ability to kill and launch Orbix servers remotely.*

- Does the product provide scripting capabilities? *Orbix itself does not provide scripting capabilities. However, it is certainly possible to use scripting languages with Orbix and many of our customers have done this.*

State Maintenance

- Can the product save object state? *Yes, there is a feature in Orbix termed a Loader that was designed specifically to allow object state to be both saved and loaded from a persistent store or database. The Loader is used to write adapters to specific databases.*

Software Distribution

- Does the product support push technology? *Yes. OrbixTalk is our asynchronous messaging product and it uses a push model to send messages from a talker to one or more listeners.*

Load Balancing

Does the product support load balancing *Yes, there are a variety of mechanisms for implementing load balancing in Orbix, all of which have little or no impact on client code:*

1. The Naming Service.

Normal use of the Naming Service involves a client querying it for an object reference that corresponds to a simple ASCII name supplied by the client. The client receives the object reference from the Naming Service and invokes on it directly. If the client supplies the name of a directory (termed a context), a list of object references are returned. It is possible to override the implementation of the Naming Service so as to supply different rules as to how object references are to be returned to the client applications. Obviously, one such rule could be to return the object reference for the server which has least load. IONA has developed a design pattern for implementing a pool of servers via the Naming Service. This design pattern can be accessed on our public web site at http://www.IONA.com/Developers/Cookbook

2. Smart Proxies or Locators

An alternative to the Naming Service is to use a "Server Finder" object in conjunction with either Orbix Smart Proxies or Orbix Locators, both of which are features of the standard product. Instead of connecting to a server directly, the client interrogates the "Server Finder" object which prescribes some load balancing scheme for the system. Thus the client application's only interaction with the "Server Finder" is to ask it for an object reference to one of the "replicated" servers. However, in the client application code it is desirable to get a reference to a "replicated" server as transparently as possible.

Two techniques used for implementing this functionality are Smart Proxies and custom Locators:

- *use a Smart Proxy to define an implementation for each member operation so that the operation is handled by a "replicated" server reference supplied by the "Server Finder".*

- *use a Locator to hook into a _bind call with no host specified and again, use it to interrogate the "Server Finder" for an object reference of one of the "replicated" servers.*

3. The Orbix Object Transaction Monitor (OTM)

Orbix OTM will offer round-robin load balancing on both an intra-host and a cross-host basis. Intra-host load balancing is achieved by notifying the ORB on server registration that a particular server is to be serviced by N server instances. The ORB will then allocate incoming requests between these servers on a round-robin basis. This is a simple strategy, which has a straightforward information and location policy, but which nonetheless can be expected to yield significant performance [throughput] enhancements over the no-load-balancing case.

Audit Trail

- Does the product provide audit capability? *Yes, the Orbix Filter feature allows all Orbix calls to be logged.*
- Does the product provide interface to audit operating system activities? *No.*
- Does the product provide interface to Sybase audit capabilities? *No.*

Error Management

- Does the product provide error handling capability? *Yes, the CORBA specification provides a very extensive set of exceptions that can be raised by CORBA calls. Further, CORBA supports the ability for developers to create user-defined exceptions as they see fit.*
- Does the product provide interface to operating system errors? *No.*
- Does the product provide interface to Sybase errors? *No.*

CORBA/Non-CORBA SERVICES

Life Cycle Service

- Does the product provide operations for creating, copying and deleting objects? *No. However, we are currently evaluating customer interest in further CORBAservices and this is probably high on the list. In the meantime it is possible to provide much of the functionality in that specification but work is required in the part of the developer.*
- Is the product life cycle service CORBA 2.0 compliant?

Persistent Service

- Does the product provide an interface for storing objects on a variety of storage servers? *No. The Persistence Service is currently undergoing a second revision within the OMG and we await the outcome of that work. In the meantime we have specific adapters to existing databases.*
- Is the product persistent service CORBA 2.0 compliant?

Naming Service

- Does the product allow objects on the bus to locate other objects by name? *Yes, we have a full implementation of the CORBA Naming Service.*
- Is the product naming service CORBA 2.0 compliant? *Yes, it is fully compliant with the CORBA specification.*

Event Service

- Does the product allow objects to register and unregister dynamically for events? *Yes, this functionality is supported. Also, our implementation is based on IP Multicast technology making it very fast and scalable.*

- Is the product event service CORBA 2.0 compliant? *Yes.*

Concurrency Control Service

- Does the product provide concurrency control service? *This is not currently available as a separate product but is implemented as part of (and bundled with) the Object Transaction Service.*

- Is the product concurrency control service CORBA 2.0 compliant? *Yes.*

Transaction Processing Service

- Does the product support some means of transaction management? *Yes, we have implemented the CORBA Object Transaction Service in conjunction with Transarc. We also have another implementation of the OTS which was done by Groupe Bull in France.*

- Is the product transaction service CORBA 2.0 compliant? *Yes, both implementations are compliant.*

- If product does not support transaction service, does it integrate with a transaction processing product? *Orbix is also integrated with Transarc's Encina TP Monitor.*

Relationship Service

- Does the product provide operations to dynamically create associations between objects that know nothing of each other? *This service is not currently supported.*

- Is the product relationship service CORBA 2.0 compliant?

Externalization Service

- Does the product provide a standard way of getting data into and out of an object? *This service is not supported.*

- Is the product externalization service CORBA 2.0 compliant?

Query Service

- Does the product provide query operations for objects? *This service is not supported.*

- Is the product query service CORBA 2.0 compliant?

Security Service

- Does the product provide security service? *We are in the beta stage of our implementation of the CORBA security service. This will implement level 1 of the specification.*

- Is the product security service CORBA 2.0 compliant? *Yes.*

- Does the product security service provide identification and authentication? *Yes.*

- Does the product security service support privilege delegation? *Yes.*

- Does the product security service support transaction authorization? *Yes.*

- If product does not support security service, does it integrate with a other Security Package - such as a DCE based one? *Our initial implementation of the Security Service will actually be built on top of DCE Kerberos.*

- Can the product integrate with a system that provides a security component? *Yes. If a customer does not wish to avail of the CORBA Security Service the Filter feature of Orbix allows other security mechanisms to be integrated. Many of our customers have previously done this.*

Licensing Service

- Does the product provide operations for metering the use of objects? *We currently do not support this service.*

- Is the product licensing service CORBA 2.0 compliant?

Properties Service

- Does the product provide operations to dynamically associate properties with object state? *We currently do not support this service.*

- Is the product properties service CORBA 2.0 compliant?

Timing Service

- Does the product provide operations for defining and managing time triggered-events? *We currently do not support this service.*

- Is the product timing service CORBA 2.0 compliant?

Trader Service

- Does the product provide means for objects to publicize their services and bid for jobs? *Yes, we are currrently in the beta stage of implementing the CORBA Trader Service.*

- Is the product trader service CORBA 2.0 compliant? *Yes.*

Collection Service

- Does the product provide interfaces to generically create and manipulate the most common collections.? *We currently do not support this service.*

- Is the product collection service CORBA 2.0 compliant?

DEVELOPMENT ISSUES

Development environment

- How easy is development effort using this product? *Orbix was designed to be easy to use for programmers and our customers have found this to be the case. From the point of view of the developer most of the code seems to be local programming. Orbix removes the complexity of network programming from the developer.*

- Is distributed object development using this product easily integrated with an open, integrated development environment? *Orbix imposes no restrictions on the use of a development environment. Once the IDL file is compiled the resulting generated code is simply included with the developers code, who is then free to use his normal development.*

- Do any language development environments generate IDL compatible with this product? *Yes, some of the current OOAD tools such as Rational Rose and Paradigm Plus generate CORBA compliant code which by default is compatible with Orbix.*

- Does the product integrate with an analysis/design tool? *See the answer to the previous question.*

Debugging Aids

- Does the product provide method tracing at any granularity? *Yes, filters can be used to monitor all the messages coming in for any method.*

- Can the product log method / messages to a log file? *Yes. Again, the use of filters allow logging to file of messages incoming for methods.*

- Can the product trace daemon activation?

- Can the product trace the instantiation and destruction of objects in the server? *Yes, Orbix has a mechanism called a loader that can allow code to be called when objects are created or destroyed.*

BUSINESS ISSUES

Technical Support

- Does the vendor provide a dedicated hot-line phone service? *IONA provides dedicated e-mail and fax support*

- Does the vendor provide electronic mail and fax service of support request? *Yes*

- Does the vendor provide World Wide Web support? *No*

- Does the vendor provide on-site support advise/troubleshooting? *Yes. IONA has a seasoned team of consultants who can provide various types of on-site assistance. Contact Steve Mosca at 800-672-4948 for more information.*

- Does the vendor provide extended service agreement? *No*

- Is the vendor responsive? *Yes. IONA has a dedicated team of experienced customer engineers whose goal is to respond to customer requests as quickly as possible.*

- Are support responses to customer questions accurate? *Yes. On the whole, our customers have been very pleased with their responses.*

- Is consulting help on the product available? *Yes. Contact Steve Mosca.*

- Does the vendor response to customer problems effectively (urgent patches)? *Yes. IONA is first and foremost a customer driven company. Many of the patches we have put out have been in direct response to customer requests.*

Viability of the vendor

- Is the vendor well capitalized? *Yes. IONA has no debt.*

- Does the vendor have seasoned management staff? *Yes. Our management staff brings with it many years of experience in the software industry.*

- Does the vendor have sufficient technical staff to evolve and support the product? *Yes.*

- Can the vendor respond effectively to changing needs of the industry or of its key customers? *Yes. As mentioned above, IONA's primary focus is on the success of its customers. We have a seasoned team of executives whose sole responsibility is to keep abreast of changes in the industry for the purpose of keeping our products superior.*

Dedication of the vendor to the product

- Is the product part of the vendor's long-term, strategic direction? *Yes. Unlike many of our competitors, the ORB is our core business, not a secondary product. IONA intends to maintain its ocus on being the best ORB provider.*

Strategic alliance partners

- Has the vendor formed alliances with significant third-party providers? *Yes.*

MARKET ACCEPTANCE

Maturity

- When was the product introduced? *1991*
- How often was the product updated? *Typically, the product is upgraded one to two times a year.*

Market Share

- What is the current market share? *Approximately 55%*
- What is the size of the current installed base? *16,000*
- What is the largest current installation of the product? *Boeing is currently our largest user. They are using Orbix to tie together all of their software systems used to design the 777.*
- What is the current market share relative to competitors? *IONA holds the largest market share.*

TRAINING

- Does vendor provide development training? *Yes*
- Does vendor provide maintenance training? *N/A*
- Does vendor provide administrative training? *N/A*
- Can unscheduled training be arranged? *Yes*

COST

- Is product currently on GSA schedule? *No. However, many government agencies including NASA have purchased Orbix open market.*

- If not, does vendor provide not-for-profit pricing? *No, but we will provide discount pricing.*
- Is the product suite competitively priced to deploy for ASAC? *Yes.*
- For a 5 servers and 40 clients ASAC AE system, what is the product suite price range? *This would depend on the platforms used, compiler, threading model, etc.*
- Is the extended service support included in price? *No.*
- If not, what is the extended service support cost? *N/A*
- Is training included priced? *No.*
- If not, what is training cost? *$1990 or a public course, $16,000 for an onsite (12 ppl max.)*
- What is the price for technical support? *$750 per developer licencse for UNIX single threaded, $975 for multi-threaded. $400 for NT.*
- What is the cost of upgrades relative to current version cost? *Minor upgrades are included in the cost of support.*

RESPONSE FROM VISIGENIC CORPORATION

2. ORB Evaluation Criteria

2.1 INTRODUCTION

As indicated earlier, the ABCD infrastructure and applications will be delivered in a phased fashion beginning with a managed set of pilot projects. Therefore, only a subset of the promised CORBA services will be required for the initial CORBA-enabled deployment of ABCD. In addition, we are prepared to deliver our own service implementations, if needed, either on a temporary or permanent basis, if suitable commercial implementations are not available.

This RFI, therefore, combines a list of immediate must-have requirements, as well as more general, longer-term evaluation criteria. Given the relative immaturity of the ORB market and associated products, we do not believe that a mere feature checklist or comparative matrix is particularly meaningful. A more helpful response will attempt to provide solutions to ABCD's requirements centered around the vendor's ORB offering.

First and foremost, our goal is to determine that a viable solution is immediately available and to identify the vendors that can help deliver that solution. Second, we will look closely at how these vendors, and possibly others that are running close behind, will be able to help fulfill our longer range objectives. Understanding the vendor's ORB feature set in the context of our requirements is central to accomplishing these goals.

2.2 ESSENTIAL REQUIREMENTS

The following list describes the essential features that an object request broker (ORB) must have to be a candidate for ABCD's global infrastructure. The features listed below should be considered the necessary, but not sufficient, set of capabilities for an ORB that can support a large, enterprise-wide distributed computing endeavor.

- The ORB vendor must be a viable company, capable of providing long-term support.
- The ORB must be CORBA 2.0 compliant.
- The ORB must be present on Windows NT/95 and Sun Solaris platforms, as well as various other UNIX platforms.
- The ORB must have C and C++ bindings.
- The ORB must have a JAVA/WWW binding or gateway.
- The ORB must have an OLE/COM or DCOM bridge.
- The ORB must have multithreading capability.

- The ORB must provide suitable LifeCycle and Object Location services (e.g. Naming, Trading, and/or Factories).

- The ORB must provide an Event service.

- The ORB must facilitate server replication and fail-over.

- The ORB must have a Security service or strategy.

2.2.1 The ORB vendor must be a viable company, capable of providing long-term support:

Justification: In buying a commercial ORB product, one acquires not only the software, but a long-term relationship with a company. This company must provide support for the current product and strategic evolution of the product to meet the developing needs of the customers. If the company is not likely to be viable, the product is of limited value, no matter what its technical merit. Product/company longevity and actual mission-critical reference sites are also a good measure of a company's viability.

Response: VisiBroker for C++ was released in 1995 and was the first CORBA 2.0 compliant C++ ORB. VisiBroker for Java was released in early 1996 and was the first CORBA 2.0 ORB written entirely in Java with full Java products running on both the client and server. VisiBroker for Java and C++ are the leading ORB's today with the upcoming shipment of approximately 20 million copies of Netscape's Navigator 4.0 which will bundle VisiBroker for Java runtime version and several million copies of Netscape's SuiteSpot Servers which will bundle VisiBroker for Java and C++. In addition, other strategic commitments have been made to Visigenic VisiBroker technology including Oracle's decision to bundle VisiBroker within its NCA architecture, Novell's choice of VisiBroker for IntranetWare, and Borland's selection of VisiBroker for Java to bundle with JBuilder. Reference contacts from Visigenic partners will be available to ABCD upon request. Visigenic Software is headquartered in San Mateo, California and became a publicly traded company in August 1996. Revenue for the first nine months of fiscal 1997 increased 248% from the comparable period of the prior year to $11.5 million. Revenue for the third quarter of fiscal 1997 was $4.8 million, representing a 336% increase over revenue in the comparable period of 1996.

2.2.2 The ORB must be CORBA 2.0 compliant:

Justification: Adherence to the CORBA 2.0 standard promises interoperability between independently developed applications across heterogeneous networks of computers. The CORBA standard has wide adoption by 700+ companies. This extensive support gives some assurance of (eventual) interoperability of CORBA ORBs and plug-and-play capability amongst conforming products.

Response: VisiBroker for C++ is CORBA 2.0 compliant. VisiBroker for Java 3.0 will be the first Java ORB that conforms to the new IDL-to-Java mapping. As soon as the CORBA conformance tests are finalized we will work diligently to ensure that VisiBroker complies.

2.2.3 The ORB must be present on Windows NT/95 and Sun Solaris platforms, as well as various other UNIX platforms:

Justification: ABCD's system currently supports Windows NT/95 and Solaris clients, as well as Solaris servers. In addition, we will require the ability to deploy Windows NT and other UNIX branded servers. Direct ORB support for Windows 3.1 would be highly advantageous; however, we are willing to consider other options that provide Windows 3.1 integration.

Response: VisiBroker runs on a variety of operating systems, including Windows NT, Windows 95, and a large number of UNIX platforms, including Sun Solaris. Below is a summary of all the platforms VisiBroker runs:

VisiBroker for C++ 2.1

Platform Vendor	Architecture	OS/Version	Compiler
Sun	SPARC	Solaris 2.4, 2.5, 2.5.1	SPARCworks C++ 4.0.1, 4.1
		SunOS 4.1, 4.1.4	
Microsoft	Intel/Win32	Window95, NT 3.51, 4.0	Visual C++ 4.1, 4.2 Watcom C++ 11.0 (TBD)
HP	PA-RISC	HP-UX 10.01, 10.10, 10.20	C-front A.10.22 and later [ANSI] aC++ (TBD)
IBM	RS6000	AIX 4.1, 4.2	C Set++ 3.1
SGI	MIPS	IRIX 6.2	MIPSpro C++ 7.1
Digital	Alpha	Digital UNIX 3.2f	DEC C++ 5.1, 5.5

VisiBroker for C++ 3.0

Platform Vendor	Architecture	OS/Version	Compiler
Sun	SPARC	Solaris 2.5, 2.5.1	SPARCworks C++ 4.1, 4.2
Microsoft	Intel/Win32	Window95, NT 3.51, 4.0	Visual C++ 4.1, 4.2, 5.0 Watcom C++ 11.0 (TBD)
HP	PA-RISC	HP-UX 10.10, 10.20 (GA + 6 weeks)	C-front A.10.22 [ANSI] aC++
IBM	RS6000	AIX 4.1, 4.2 (GA + 6 weeks)	C Set++ 3.1 (or latest)
SGI	MIPS	IRIX 6.2 (TBD)	MIPSpro C++ 7.1 (or latest)
Digital (TBD)	Alpha	Digital UNIX 3.2 and 4.0	DEC C++ 5.5 (or latest)

***VisiBroker for Java 1.2—JDK 1.0.x compatible (not JDK 1.1)**

Platform Vendor	Architecture	OS/Version
Sun	SPARC	Solaris 2.4, 2.5, 2.5.1
Microsoft	Intel/Win32	Window95, NT 3.51, 4.0
HP	PA-RISC	HP-UX 10.01, 10.10
IBM	RS6000	AIX 4.1
SGI	MIPS	IRIX 6.2
Digital (TBD)	Alpha	Digital UNIX 3.2 (not planned)

*requires OSAgent running on some server

VisiBroker for Java 3.0

*Platform Vendor	Architecture	OS/Version
Sun	SPARC	Solaris 2.5, 2.5.1
Microsoft	Intel/Win32	Window95, NT 3.51, 4.0
HP	PA-RISC	HP-UX 10.10, 10.20 (Q3)
IBM	RS6000	AIX 4.1, 4.2 (Q3)
SGI	MIPS	IRIX 6.2 (Q3)
Digital (TBD)	Alpha	Digital UNIX 4.0 (not planned)

* plus any other platform with a JVM 1.0 or Higher, OSAgent not required

2.2.4 The ORB must have C and C++ bindings:

Justification: This is almost a given because all the commercial ORBs have a C++ binding for the current version of the CORBA standard. The previous version of the standard provided only a C binding; ORBs compliant with this earlier standard would require developers to manipulate C structures rather than objects, eliminating many of the benefits of object technology and increasing code maintenance effort and cost. However, ORBs that offer both bindings facilitate integration with various RAD presentation tools (e.g. Visual Basic), as well as legacy applications, thereby providing for true enterprise interoperability.

Response: Visigenic's VisiBroker for C++ supports C++, Visigenic's VisiBroker for Java supports Java and

Smalltalk support is provided by DNS. VisiBroker and the DNS ORB can interoperate.

2.2.5 The ORB must have a JAVA/WWW binding or gateway:

Justification: The growing popularity of distributed computing applications for the Internet or for enterprise-wide Intranets requires that an ORB provide such a gateway to its backend capability. While a script activated by an HTML interface may serve as a bridge technology, a script is inconsistent with the dynamic, downloadable nature of current Web technology, epitomized by JAVA.

Response: The IIOP GateKeeper allows Java clients (built using VisiBroker applets) and servers running with any Java 1.0 or better compatible browser to be transparently available to other CORBA objects, even when a firewall is in place.

For security reasons, web browsers are only allowed to communicate with the host specified in the URL (the sandbox model). By acting as an IIOP proxy, the GateKeeper allows client applets to effectively communicate with any other host. Each client operation request, targeted to an object running on another host, is sent to the GateKeeper, which forwards the request to the appropriate server and returns the response.

If there are Firewalls present the GateKeeper also supports HTTP tunneling, which enables a client to communicate

through a firewall that only allows HTTP communication. If a client attempts to bind to an object and the IIOP-style communication fails, the IIOP request will automatically be encapsulated in an HTTP-style request. The GateKeeper, upon receiving an encapsulated IIOP request, will automatically detect that HTTP tunneling is being used.

The GateKeeper supports callbacks (callback from a server to a client) A callback can occur when a client application sends an object reference to a server and, in response, the server might need to invoke an operation on the object reference. A message is then sent to the client application.

When a distributed object is exported through the Gatekeeper, a proxy in the Gatekeeper is *dynamically created* and the client communicates with the proxy, instead of directly with the object. The proxy simply forwards the request to the real distributed object.

Using the VisiBroker Gatekeeper is *simple and transparent.* There are no application level APIs to the Gatekeeper, nor is it necessary to maintain any configuration files. Applications can create distributed objects and pass them around as they can within the Intranet.

2.2.6 The ORB must have an OLE/COM or DCOM bridge:

Justification: Integration with the predominant desktop computing environments - Windows 3.1, Windows NT, and Windows 95 - is a paramount concern for any integration facility. Thus, an ORB must be able to address the needs of an existing installed base of clients that utilize Windows machines while providing access to globally-distributed TBU servers.

Response: The Visigenic COM\CORBA Bridge (VisiBridge) enables a COM client application to use a CORBA server. The COM client application may be one of these types:

- an OLE Automation controller like Visual Basic, Excel, or PowerBuilder
- a custom application written either in C++ or Java, using direct calls on COM interfaces

Visigenic's COM\CORBA Bridge includes a Bridge Wizard that creates bridge objects automatically without any programming. The bridge objects that are created using the Bridge Wizard may be one of two types:

- an ActiveX control
- a standalone DCOM server

VisiBridge supports three type of bridging models:

1. ActiveX control bridge object, as a DLL, is an in-process server. The DLL is loaded into the COM client application's process. The bridge object as an ActiveX control communicates with the CORBA server using IIOP. IIOP (Internet Inter-ORB Protocol) is a protocol standard which will be mandatory for all CORBA 2.0 compliant platforms.

2. The bridge object as a standalone DCOM server. The bridge object is an executable located on the same machine as the COM client application-it is not an in-process server. The COM client application communicates to the bridge object using LRPC (light-weight RPC). The bridge object communicates to the CORBA server, which is located on a remote server, using IIOP.

3. The bridge object is a standalone DCOM server located on the remote server with the CORBA server. The COM client application, on the local machine, communicates to the bridge object using DCOM. The bridge object communicates to the CORBA server using IIOP.

   Note: The bridge object in the second and third models is the same. The difference is in its location. In the second model the bridge object is on the same machine as the COM client application that makes it a COM server. In the third model, it is on a different machine than the one where the COM client is installed making it a DCOM server.

Later during 1997, Visigenic will allow for two-way bridging, CORBA objects accessing COM/DCOM objects.

2.2.7 The ORB must have multithreading capability:

Justification: Multithreading is a minimum capability for supporting large-scale applications. Multithreading capability can take advantage of multiple processors, when they exist, but can also more effectively utilize a single processor by enabling lightweight task switching. The ability to provide multithreading to systems that are natively not multithreaded (i.e., Windows 3.1) is also paramount.

Response: The following describes VisiBroker for Java and C++, thread and connect management schemes (Versions 2.0 and 3.0).

VisiBroker 2.0 Model Connections

When a 2.0 client binds to a server, a single connection is created. In a single-threaded client, all calls from that client to the same server process will reuse an existing client connection. For example, if a single-threaded client has references to two objects in the same server process, and the client makes calls on each object, both calls are made on the same connection.

However, in a multi-threaded client, a new connection is created when a bind or clone is performed in a new thread. Thus, if a client creates two threads, and in each one, binds to two objects in the same server process (as described above), two separate connections will be created. A new connection would also result if one thread performed the bind, and another thread calls clone( ) on the resulting object reference. Note that even in a multi-threaded client, a given thread can only create one connection to a given server process. Thus, binds to objects in the same server process or cloning object references will not cause new connections to be created, unless a new client thread has been created.

In the 2.0 model, multiplexing over a single connection can occur when multiple calls are made to the same server process. This means the client issues a single request and the requesting client thread blocks on the connection until a response is received. So even though requests to more than one server object, for example, are being multiplexed on a single connection, each request must complete before the next one is allowed to initiate.

Server Threads

When a connection is created to a 2.0 server, a worker thread is created on behalf of that connection. It exists as long as the connection exists, and is destroyed when the connection is destroyed. Thus, there is a one-to-one correspondence between connections to a server process, and the number of worker threads for a server. If 50 connections are created to a given server process, 50 threads will be created on their behalf, regardless of

VisiBroker 3.0 Model Connections

3.0 simplifies client connections in that all client threads in a process share a single connection to a server process. Thus, separate threads multiplex over one connection, even if they perform binds. This is not the case, of course, if the client explicitly calls clone( ). Connections in the server process can be adjusted dynamically, allowing a maximum number of connections and connection recycling of the least recently used connections. Connection recycling is handled transparently; there is no need for developers to code for this situation.

Multiplexing in the 3.0 model has been enhanced over 2.0, and as a result, calling clone ( ) and explicitly creating a new connection is rarely necessary. In 3.0, multiple requests can be initiated without intermediate responses. So over the same connection, the client can initiate several requests to a server process (presumably to multiple server objects), and as each response is received by the client, the results are handed off to the appropriate client thread. This can result in significant performance enhancement, and requires fewer connections to the server since regardless of the time it takes to service a request, a single connection will return the results as fast as separate connections would have done.

Server Threads

3.0 offers two BOAs that differ by their thread policy. The thread-per-session BOA (named "Tsession") offers the same thread policy as 2.0 (see Server Threads for 2.0 described above). The new BOA in 3.0 is the thread-pool BOA (named "Tpool"). Instead of creating threads per connection, threads are allocated as needed but, when the connection closes, threads are returned to a pool and not destroyed. Thus, threads are used on per request, resulting in better performance across fewer threads. For example, 50 clients may use far fewer than 50 threads to perform work on the server objects in the process. By default, the pool continues to grow as multiple simultaneous requests are serviced, but the size of the thread pool can be adjusted from the server at runtime.

Performance Differences Between 2.0 and 3.0—Clients

On the 2.0 client side, a client using multiple object references to the same server process uses the same connection for all calls unless it creates separate threads and performs a bind() or clone(). Thus, to achieve parallel calls, a client ends up creating new connections resulting in additIONAl resources.

In 3.0, clients multiplex requests on the same connection to a server process unless they explicitly issue a clone( ). Since the multiplexing behavior allows client requests to be serviced out of order, the performance of this system looks to the client as though each request was issued on a separate connection. For best performance, the server process should be using the thread pool BOA so that it can service each request in a separate thread if necessary. If the server uses the thread-per-session BOA, since only one connection is created, a single thread will service all requests and the client multiplexing will not be useful.

Performance Differences Between 2.0 and 3.0—Servers

On the 2.0 server side, servers use the thread-per-session model, which means that they consume threads based on the number of connections. Multiple requests that come in on the same connection are serviced serially, because only one thread is associated with that connection. And when clients/connections are idle, their associated threads are idle.

In 3.0, the thread-pool BOA creates threads based on simultaneous requests rather than the number of outstanding connections. The number of threads can be adjusted dynamically, giving the server process maximum flexibility. In a typical scenario, a server has several clients connected to it and is servicing requests on an occasIONAl basis. Since the number of threads are based on the number of concurrent requests, fewer threads are allocated in the server, and these threads are busier more often since they aren't tied to a specific connection.

Multi-threading and Windows 3.1

A multi-threaded server can implement a complex Windows user interface either directly on the Win32 API or using MFC.

A key point in building a multi-threaded server with an MFC-based Windows user interface is that only certain threads may do user interface update. Within an MFC application, either the main application thread or a CWinThread-derived class that the application created may do user interface updates. These restrictions are because MFC threads contain thread-local storage important in doing user interface updates. Performing a user interface update from a non-MFC thread causes errors because the system does not have the required local storage within the thread.

Because VisiBroker for C++ creates a worker thread for each incoming connection, these threads are not MFC threads. Such a worker thread cannot perform user interface updates directly.

It is straightforward to interface between VisiBroker threads and MFC threads. The VisiBroker thread can post an invalidate message to the window to update. The message may contain either the needed information for update or the two threads may use a common object or data structure to pass information.

2.2.8 The ORB must provide suitable Lifecycle and Object Location services (e.g., Naming, Trading, and/or Factories):

Justification: Location-independent transaction routing is a fundamental capability of any enterprise-scale ORB. Trading provides the richest set of features to unambiguously identify and utilize objects. Nonetheless, Lifecycle services are important in their own right, and various combinations of Lifecycle and Naming can provide comparable functionality.

Response: The OMG LifeCycle spec defines a standardized set of LifeCycle operations such as move and copy that an object may support and conventions regarding object factories that an application can conform to. There is not, per se, anything that can be implemented as a service library that is linked into a server because the implementation of the lifecycle interfaces are usually application specific or at least involves a complex interaction of a number of services and ORB functions that are not easily encapsulated.

Including the client stubs as part of the VisiBroker 3.0 Developer products is under consideration. Note that providing these stubs is primarily a developer convenience since the developer can generate them anyway by taking the lifecycle IDL definitions and compiling them using the IDL compiler.

VisiBroker's Smart Agent (OSAgent) is a dynamic, distributed directory service that provides facilities for both client applications and object implementations. When a client application invokes the bind method on an object, the OSAgent locates the specified implementation and object so that a connection can be established between the client and the implementation. Object implementations register their objects with the OSAgent so that client applications can locate and use those objects. When an object or implementation is destroyed, the OSAgent removes them from its list of available objects.

An OSAgent may be started on any host. To locate an OSAgent, client applications and object implementations send a broadcast message, and the first OSAgent to respond will be used. Once an OSAgent has been located, a point-to-point UDP communication is established for registration and look-up requests. The UDP protocol is used because it consumes fewer network resources than a TCP connection. All registration and locate requests are dynamic, so there are no required configuration files or mappings to maintain.

If you run more than one instance of the OSAgent on a local network and one of those agents becomes unavailable, all object implementations registered with that agent will be automatically re-registered with another agent. Likewise, client applications using an OSAgent that becomes unavailable will be automatically switched to another agent by VisiBroker. No special coding techniques are required to take advantage of this OSAgent fault-tolerance, as long as the OSAgent is running on more than one host on your local network.

LOCATION SERVICE COMPONENTS

The Location Service is accessible through the CORBA interface Agent. Agent methods can be divided into two groups, those that query an OSAgent for data describing instances, and those that register and unregister triggers. Triggers are a notification mechanism whose methods are defined in the Agent and TriggerHandler interfaces. Whereas queries can be employed in many ways, triggers are special-purpose, and are used by comparatively few developers.

THE AGENT

The bulk of the Agent methods constitute a simple query facility; you invoke the method that a) identifies the instances you want, and b) returns the data you want for those instances. Each Agent query method returns a sequence of one of the following:

- Object reference: You can use an object reference to invoke an instance found by the OSAgent.
- Desc: An instance description; a struct containing the instance's full description; its interface, its name, its host and port, an object reference, and whether it is running or can be activated.
- string: Host names are returned as strings.

NOTE: Earlier versions of the VisiBroker ORB used IDL interface names to identify interfaces, but the Location Service uses repository id instead. To illustrate the difference, if an interface name is

::module1::module2::interface,

the equivalent repository id is

IDL:module1/module2/interface:1.0.

Triggers

A trigger is a means of learning when a specified kind of instance becomes accessible. It is an asynchronous alternative to polling an Agent, and is typically used to recover after the connection to an object has been lost.

To register a trigger, you pass a description of the instance you want, and the TriggerHandler object you want invoked when the instance becomes available. The TriggerDesc can contain any combination of repository id, instance name, and host; the more fields you provide values for, the more particular your specification of the instance. For example, a TriggerDesc containing only a repository id matches any object that satisfies the interface; adding an instance name tightens the specification, and adding a host name tightens it further one instance.

Naming Service

Client applications normally bind to objects they wish to use by specifying an interface name. The naming service allows an object implementation to associate one or more arbitrary names with each of the objects it offers. Client applications can then use the naming service to locate these ORB objects by their bound name rather than by their interface name.

The naming service organizes names in much the same way that files are organized in a file system. A name is always bound to an object within a naming context, which is similar in concept to a directory in a file system. Each naming context contains a list of named objects, which is similar to the way a directory contains files in a file system. Naming con-texts may also contain other naming contexts, just as a directory may contain sub-directories. The result is a hierarchical namespace that can be traversed to locate a desired object. The naming system has no concept of a root naming context, unlike a file system, which always has a root directory. For example, one set of naming contexts could be used by a research group and another set could be used by a manufacturing group. Each of these groups could have what they consider to be a root naming context. The naming contexts of these two groups could subsequently be federated into a company-wide naming context, resulting in arbitrarily nested naming contexts with no global root context.

The Visigenic Naming Service is available in Java or C++.

2.2.9 The ORB must provide an Event service:

Justification: An event service provides a flexible means to allow separate processes to communicate. In the absence of a standard messaging delivery option—to complement the standard synchronous RPC model—an event service enables the development of client asynchrony. In addition, an event service provides the essential foundation for the development of streaming interfaces, both for large multimedia content as well as streaming audio or video, as well as the delivery of PUSH functionality.

Response: Visigenic offers a CORBA 2.0 compliant Event Service. (The Event Service package provides a facility that de-couples the communication between objects. It provides a supplier-consumer communication model that allows multiple supplier objects to send data asynchronously to multiple consumer objects through an event channel. The supplier—consumer communication model allows an object to communicate an important change in state, such as a disk running out of free space, to any other objects that might be interested in such an event.)

The event channel is both a consumer of events and a supplier of events. The data communicated between suppliers and consumers are represented by the Any class, allowing any CORBA type to be passed in a type safe manner. Supplier and consumer objects communicate through the event channel using standard CORBA requests.

The event service provides both a pull and push communication model for suppliers and consumers. In the push model, supplier objects control the flow of data by pushing it to consumers. In the pull model, consumer objects control the flow of data by pulling data from the supplier. The EventChannel insulates suppliers and consumers from having to know which model is being used by other objects on the channel. This means that a pull supplier can provide data to a pull consumer.

Visigenic Event Services is available in C++ and Java.

2.2.10 The ORB must facilitate server replication and fail-over:

Justification: Continuous availability of data (with good performance) and the ability of a system to heal itself in recovering from errors are two necessary features in any mission-critical system. It is not feasible to implement adequate fault resilience without some vendor-provided facilities.

Response: Object implementation fault tolerance for objects is implemented by simply starting instances of those objects on multiple hosts. The ORB, via the OSAgent, will detect the loss of the connection between the client application and the object implementation and the ORB will automatically attempt to establish a connection with another instance of the object implementation. The client can continue invoking methods on the object without being concerned that a new instance of the object is being used.

2.2.11 The ORB must have a Security servlce or strategy:

Justification: Security is a key element of the ABCD infrastructure. Each TBU will have differing security requirements; however, authentication and access control are required across the board, and integrity and privacy will be essential to a number of our early projects. Therefore, the ORB vendor must provide some form of security option, preferably conformant to or with a clear migration path towards conformance with the CORBA 2.0 Security Service specification. Initially, access control and auditing will require Level 1 compliance with the security specification; ultimately, support at Level 2 of the specification will become an essential requirement. In addition, the encryption implementation should support both full strength encryption for domestic use, and weakened encryption for export. Also, authentication and encryption methods should be changeable via shared object or dll substitution rather than full product upgrade.

Response: Visigenic currently is ready to release phase one of security with VisiBroker 3.0; Secure IIOP (OMG compliant). A full implementation of the CORBA 2.0 specification will be available later in 1998.

The Secure BOA (SBOA) uses the industry-standard Secure Socket Layer (SSL) protocol to establish secure connections between clients and servers. The SSL protocol provides clients and servers with

- privacy: Data passed between them is encrypted.
- integrity: Checksums in the data detect accidental or malicious corruption.
- authentication: A client is assured that a server is not an impostor; optIONAlly, a server can require the same assurance of its clients.

In addition to implementing SSL privacy, integrity, and authentication, the SBOA has these features:

- Existing servers and clients can be made secure without altering their core code; at most, their initialization code has to be changed to work with the SBOA. Clients that don't require server authentication need no changes to work with servers that use the SBOA.

- SBOA-based servers can identify client users for logging or for restricting execution of sensitive operations to authorized users.

- Secure connections are orthogonal to threads and other VisiBroker facilities; choosing security does not mean giving up something else.

- The SBOA's implementations of credential checking and encryption can be customized by developers.

In addition to the SSL and the SBOA, you can utilize VisiBroker's interceptors. You can add to or alter the data passed between clients and servers; for example, you can add transaction information or encrypt a credit card number.

2.3 General Criteria

This section describes general criteria for evaluation of an ORB and its vendor. Questions in each category that address one of the critical requirements are noted with the star symbol (*). The following categories will be addressed:

- Standards Compliance

- Platforms and Bindings

- Scalability, Availability, and Fault Resilience

- Performance and Resource Utilization

- Integration and Interoperability

- OMG Services

- Development Issues

- Management

- Usability and Customizability

- Business Issues

- Pricing Issues
- Third-Party Products
- Case Studies.

2.3.1 Standards Compliance

The most significant standard that we are evaluating an object request broker against is its conformance to the CORBA standard (now at level 2.0). Most commercial object request brokers claim CORBA standard compliance, but the extent of the compliance may vary from product to product.

In particular:

- Does the ORB support the full CORBA 2.0 Interface Definition Language (IDL)?

  Response: Yes, both ORB products fully support the CORBA 2.0 IDL specification. In fact, the Visigenic specification for Interface Definition Language (IDL) to Java language mapping has been recommended for adoption by the Object Management Group (OMG).

- Does the ORB support the full CORBA 2.0 Application Programming Interface (API)?

  Response: CORBA 2.0 has a number of components including Core, Language Mappings, Security, and Interoperability. These are separate compliance points. The main APIs are in the Core ORB, therefore making VisiBroker fully compliant.

- Does the ORB support all the major components of the CORBA spec - BOA, IR, DII, DSI, and object contexts?

  Response: Yes, VisiBroker supports the Basic Object Adapter (BOA), the Interface Repository (IR), the Dynamic Invocation Interface (DII) and the Dynamic Skeleton Interface (DSI). VisiBroker 3.0 for C++ and Java supports all of these.

- Does the ORB support the Internet Inter-ORB Protocol (IIOP)?

  Response: Yes, the Visigenic Orb's; VisiBroker for C++ and VisiBroker for Java support IIOP as its native protocol. Visigenic was the first company to fully support IIOP in its ORB products.

In addition, some non-standard features or services may be reviewed favorably if they are likely to be incorporated into or otherwise affect the evolution of new CORBA standards. The central criterion is the degree to which a currently proprietary or non-standard feature facilitates, rather than hinders, the development of standard-based solutions.

2.3.2 Platforms and Bindings

A critical factor in ORB selection is whether the ORB resides on the platforms that one intends to utilize. A related factor of similar importance is the issue of language bindings for a given platform. Currently three language bindings have been accepted by the OMG: C, C++, and Smalltalk. An RFP for a COBOL binding has been issued. Some vendors have jumped ahead of the standard and have released COBOL, JAVA, and ADA bindings, anticipating the demand for these bindings from customers.

Please provide a summary of currently supported and soon-to-be supported platforms and bindings. Please include software version information for existing support, as well as target availability dates for expected support.

VisiBroker for C++ 2.1

Platform Vendor	Architecture	OS/Version	Compiler
Sun	SPARC	Solaris 2.4, 2.5, 2.5.1	SPARCworks C++ 4.0.1, 4.1
		SunOS 4.1, 4.1.4	
Microsoft	Intel/Win32	Windows '95, NT 3.51, 4.0	Visual C++ 4.1, 4.2 Watcom C++ 11.0 (TBD)
HP	PA-RISC	HP-UX 10.01, 10.10, 10.20	C-front A.10.22 and later [ANSI] aC++ (TBD)
IBM	RS6000	AIX 4.1, 4.2	C Set++ 3.1
SGI	MIPS	IRIX 6.2	MIPSpro C++ 7.1
Digital	Alpha	Digital UNIX 3.2f	DEC C++ 5.1, 5.5

VisiBroker for C++ 3.0

Platform Vendor	Architecture	OS/Version	Compiler
Sun	SPARC	Solaris 2.5, 2.5.1	SPARCworks C++ 4.1, 4.2
Microsoft	Intel/Win32	Windows '95, NT 3.51, 4.0	Visual C++ 4.1, 4.2, 5.0 Watcom C++ 11.0 (TBD)
HP	PA-RISC	HP-UX 10.10, 10.20 (GA + 6 weeks)	C-front A.10.22 [ANSI] aC++
IBM	RS6000	AIX 4.1, 4.2 (GA + 6 weeks)	C Set++ 3.1 (or latest)
SGI	MIPS	IRIX 6.2 (TBD)	MIPSpro C++ 7.1 (or latest)
Digital (TBD)	Alpha	Digital UNIX 3.2 and 4.0	DEC C++ 5.5 (or latest)

***VisiBroker for Java 1.2—JDK 1.0.x compatible (not JDK 1.1)**

Platform Vendor	Architecture	OS/Version
Sun	SPARC	Solaris 2.4, 2.5, 2.5.1
Microsoft	Intel/Win32	Windows '95, NT 3.51, 4.0
HP	PA-RISC	HP-UX 10.01, 10.10
IBM	RS6000	AIX 4.1
SGI	MIPS	IRIX 6.2
Digital (TBD)	Alpha	Digital UNIX 3.2 (not planned)

*Requires OSAgent running on some server.

VisiBroker for Java 3.0

*Platform Vendor	Architecture	OS/Version
Sun	SPARC	Solaris 2.5, 2.5.1
Microsoft	Intel/Win32	Windows '95, NT 3.51, 4.0
HP	PA-RISC	HP-UX 10.10, 10.20 (Q3)
IBM	RS6000	AIX 4.1, 4.2 (Q3)
SGI	MIPS	IRIX 6.2 (Q3)
Digital (TBD)	Alpha	Digital UNIX 4.0 (not planned)

*Plus any other platform with a JVM 1.0 or higher, OSAgent not required.

Also, please indicate if there are any limitations applicable to support of a particular platform or binding. In particular, please specify what OS versions and compiler versions are supported by each version of the ORB, and state if there are any interoperability limitations between different ORB versions on different platforms.

2.3.3 Scalability, Availability, and Fault Resilience

Scalability, availability, and fault resilience are essential for ABCD to deliver a global applications infrastructure. Virtually limitless scalability is one of the core promises of ORBs. Of course, continuous availability and automatic error recovery are essential to any truly large-scale system. To determine if an ORB can deliver these capabilities, we pose the following questions:

2.3.3.1 Scalability

- Threads are a light-weight means of supporting concurrency. Does the ORB support threads and threadsafe runtime libraries?*

  Response: VisiBroker provides two sets of libraries; a single-threaded library and another library that is thread-safe and re-entrant. If you use the multi-threaded version of the library, the VisiBroker core will automatically use threads for its internal processing, resulting in more efficient request management. For applications that never intend to use threads, the single-threaded library offers a set of interfaces similar to the multi-threaded library. This allows you to use the single-threaded library initially and then re-compile with the multi-threaded in the future with virtually no interface changes.

  All code within a server that implements an ORB object must be thread-safe if it is to use VisiBroker's multi-threaded library. You must take special care when accessing a system-wide resource within an object implementation. For example, many database access methods are not thread-safe. However, Visigenic has threadsafe ODBC drivers for Oracle, Sybase, and Informix.

- Services such as interface repositories and trading services need to utilize industrial—strength substrates in order to scale. Can the interface repository or trading service implementations be replaced with an industrial strength substrate, such as a database?

  Response: The trader service can have different backend databases via ODBC and more directly for specific data stores. The interface respository is currently tied to simple file storage but the scalability comes from the fact that the IR server can run on as many machines as necessary. IR is only needed for apps that use the DII. The ORB itself does not need the IR to run. Visigenic is considering having the IR use a DBMS for its repository in future releases.

- Can current users of the ORB testify to its ability to scale-up to an enterprise-wide solution? Is there anecdotal data indicating such scale?

  Response: VisiBroker for Java and C++ are the leading ORB's today with the upcoming shipment of approximately 20 million copies of Netscape's Navigator 4.0 which will bundle VisiBroker for Java runtime version and several million copies of Netscape's SuiteSpot Servers which will bundle VisiBroker for Java and C++. In addition, other strategic commitments have been made to Visigenic VisiBroker technology including Oracle's decision to bundle VisiBroker within it's NCA architecture, and Borland's selection of VisiBroker for Java to bundle with JBuilder. Reference contacts from Visigenic partners will be available to ABCD upon request.

- Lacking such data is there compelling evidence from performance monitoring, stress testing, and capacity modeling that indicates the ORB will scale?

Response:

Connection Management: VisiBroker offers superior connection management. Should a client request connections to multiple objects residing on the same server, the requests are multiplexed over a single network connection. Developers can also streamline system traffic by limiting the number of connections for each server object using the *connection pool* feature that will be available in the VisiBroker 3.0 time-frame. When the connection limit is reached, connections that have gone the longest without use are broken before new connections are created. This allows large numbers of clients to access the service provided by a given object.

SmartBinding: SmartBinding ensures that the optimum transport mechanism is chosen whenever a client binds to a server object. If the object is local to the client process, the client performs a virtual function call. If the object resides in a different process on the same host, the client uses an optimized interprocess communication mechanism. If the object resides on a different host, the client uses IIOP over the network.

Dynamic Object Creation: Dynamic object creation further enhances scalability and performance. Objects are registered with VisiBroker's Object Activation Daemon, which run on each machine that hosts server objects. If a client attempts to bind to an object that is not currently running, the activation daemon can start an instance of the object. Objects can also be started manually. Use of the Object Activation Daemon is optIONAl. VisiBroker for Java will provide an OAD in version 3.0.

- Does the ORB provide factory mechanisms for natural decomposition of applications?

Response: VisiBroker provides some very sophisticated hooks for garbage collection. The Visigenic ORB/BOA must manage the activation/deactivation of objects that are under its control and provide the illusion to clients that *all* objects are active all the time even though they are not. There are many strategies an ORB/BOA may choose for providing this illusion. For example, an object can explicitly issue a BOA::deactivate_obj call to let the ORB know its OK to deactivate the object.

You create client or server interceptor instances indirectly, by means of another class called a factory. For client interceptors you derive a subclass from VISChainClientIntercepFactory. Your implementation must override the base class's create() method, returning an instance of your client interceptor, which is itself derived from ISClientInterceptor. The base class also defines static add() and remove() methods, which add/remove a factory to/from a chain. As part of the client's initialization, create an instance of your factory in the usual way, then invoke VISChainClientIntercepFactory:: add(). Whenever the client binds to a new CORBA object, the ORB will invoke your factory's create() method, which will return the interceptor for that object. Server interceptor instances are created like client interceptor instances. You provide a factory implementation by deriving from VISChainServerIntercepFactory,

overriding the create() method to return an instance of your derivation of VISServer-Interceptor.

During server initialization, you create an instance of your factory in the usual way, then invoke VISChainServreIntercepFactory::add(). Whenever a new connection is made, the ORB will call your factory's create() method which will return the interceptor for that connection. You do not need to release or free interceptors; the ORB cleans them up when their process/object/connection goes away.

2.3.3.2 Availability

- Does the ORB support replication of servers (to increase throughput by load-balancing data access)?

  Response: Multiple OSAgents running in the same LAN locate each other and partition the name space among themselves automatically. If multiple instances of server objects are running, the OSAgent will perform load-balancing in a round-robin fashion. Object migration is the process of terminating an object implementation on one host and then starting it on another host. Object migration can be used to provide load balancing by moving objects from overloaded hosts to hosts that have more resources or processing power. Object migration can also be used to keep objects available when a host has to be shut down for hardware or software maintenance.

- Are the load balancing algorithms fixed or configurable?

  Response: The Location Service is a general-purpose facility for monitoring instances. It can be used, for example, for load balancing. Suppose that replicas of an object are deployed on several hosts. A bind interceptor can maintain a cache of the host names that offer a replica, and each host's recent load average. The interceptor can use the Location Service to update its cache by asking the Location Service for the hosts currently offering instances of the object, and then query the hosts to get their load averages. The interceptor can return an object reference for the replica on the host with the lightest load.

  Interceptors, which are extensions that the VisiBroker ORB invokes at particular points in its processing. Interceptors, in other words, are functIONAlly similar to "hooks" or "delegates" provided by some other systems. You implement an interceptor by deriving a class from a VisiBroker ORB base class, and overriding the methods that you want invoked. You can use interceptors for diverse functions, including logging, access control, request forwarding, and load balancing.

2.3.3.3 Fault Resilience

- Does the ORB support fail-over (standby server) for fault recovery?*

Response: If you run more than one instance of the OSAgent on a local network and one of those agents becomes unavailable, all object implementations registered with that agent will be automatically re-registered with another agent. Likewise, client applications using an OSAgent that becomes unavailable will be automatically switched to another agent by VisiBroker. No special coding techniques are required to take advantage of this OSAgent fault-tolerance, as long as the OSAgent is running on more than one host on your local network.

You can provide object implementation fault tolerance for objects by simply starting instances of those objects on multiple hosts. The ORB will detect the loss of the connection between the client application and the object implementation and the ORB will automatically attempt to establish a connection with another instance of the object implementation. The client can continue invoking methods on the object without being concerned that a new instance of the object is being used.

- Are there facilities in the ORB that allow graceful modification of the interface as defined between client and server (covered in the ORB Interface Type Version Management RFP)?

  Response: The Interface Type Version Management RFP did not actually result in a specification of anything. A certain company (Sun) made the point that interface versioning can be handled perfectly well by the use of subtyping and a module version numbering scheme. This argument carried the day (Visigenic actually pulled together this response for Sun).VisiBroker can be used in this way, therefore making it CORBA-compliant.

2.3.4 Performance and Resource Utilization

ABCD's primary objective is to deliver a transaction-oriented infrastructure. Multimedia streaming applications will also be required; however, the primary performance context will involve request/response transactIONAl usage. Therefore, it is crucial that transactIONAl performance of the ORB is comparable to other middleware packages. Please provide any data that will provide the necessary level of confidence that performance of the ORB will meet or exceed these expectations. Please confirm that the ORB will use the appropriate invocation mechanism (e.g., IPC for intra-host) rather than relying exclusively on inter-host transports.

Response:

Client and Server on Different Hosts: When a client requests an object that resides on a remote host, a TCP/IP connection will be established between the client and object server. The ORB will instantiate a stub (stubs are also called proxy objects) for your client to use. A method invoked on the proxy object will be written to a buffer, or marshaled, as a request and sent to the server on the remote host. The server on the remote host will unmarshal the request, invoke the desired method, and send the results back to the client.

Client and Server in a Single Process—Java: The previous discussions have assumed that object implementations have taken the form of a server process. While this is often the case, a client application and the object implementation can both be packaged inside a single process. When your client application invokes a bind in this scenario, the ORB will return a pointer to the object implementation itself. That pointer will be widened to the object type used by your client application. All methods invoked on your client's object will get called directly as Java methods on the object implementation. The ORB will be involved only during the bind process.

Client and Server on the Same Host C++: If the ORB determines that the requested object implementation resides on the local Windows host, a connection will be established between the client and server object using shared memory — only if both the client and server are multi-threaded. The ORB will instantiate a proxy object for your client to use. All methods invoked on the proxy object will be packaged as requests and sent to the server using shared memory.

If the ORB determines that the requested object implementation resides on the local UNIX host, a connection will be established between the client and server object using shared memory — only if both the client and server are multi-threaded. The ORB will instantiate a proxy object for your client to use. All methods invoked on the proxy object will be packaged as requests and sent to the server using shared memory.

Client and Server in a Single Process: The previous discussions have assumed that object implementations have taken the form of a server process. While this is often the case, a client application and an object implementation can be packaged within a single process. When your client application invokes a bind in this scenario, the ORB will return a pointer to the object implementation itself. That pointer will be widened to the object type used by your client application. All methods invoked on your client's object will be invoked as C++ virtual functions on the object implementation. The ORB will be involved only during the bind process.

Marshalling: The IIOP protocol used CDR encoding. This encoding states that the sender always sends data in senders "data format". The receiver is responsible for "conversion" if the "byte ordering" of the receiver is different. The header information in IIOP packets contain the "byte order" of the sender and the receiver has to "make it right". So two identical machines do not pay a penalty of "byte swapping".

In addition, resource requirements will affect both the cost and scalability of the system. Therefore, please indicate any known system resource usage requirements for your ORB, as well as information pertaining to efficiencies built into the ORB. For example, what is the level of socket usage, does the orb multiplex over sockets where feasible, do inactive connections get timed-out or reused?

Response: VisiBroker 3.0 simplifies client connections in that all client threads in a process share a single connection to a server process. Thus, separate threads multiplex over one connection, even if they perform binds. This is not the case, of course, if the client explicitly calls clone( ). Connections in the server process can be adjusted dynamically, allowing a maximum number of connections and connection recycling of the least recently used connections.

Multiplexing in the 3.0 model has been enhanced over 2.0, and as a result, calling clone ( ) and explicitly creating a new connection is rarely necessary. In 3.0, multiple requests can be initiated without intermediate responses. So over the same connection, the client can initiate several requests to a server process (presumably to multiple server objects), and as each response is received by the client, the results are handed off to the appropriate client thread. This can result in significant performance enhancement, and requires fewer connections to the server since regardless of the time it takes to service a request, a single connection will return the results as fast as separate connections would have done.

2.3.5 Integration and Interoperability

Integration with main frames in order to access legacy data and integration with the desktop are of paramount concern in ORB middleware. There are many legacy systems that may need to be integrated with an ORB, including VSAM and IMS systems. The desktop environment, however, is more specific. The predominant desktop computing environment—Windows 3.1, Windows NT, and Windows '95—is a paramount concern for any integration facility. Thus, an ORB must be able to address the needs of an existing installed base of clients that utilize Windows machines while providing access to enterprise servers. Questions to be addressed in evaluating the ability of an ORB to integrate with existing systems are:

2.3.5.1 Access to the Mainframe

Does the ORB support access to the mainframe?

Response: Visigenic is working with a partner to provide an MVS port of VisiBroker. Beta availability of VisiBroker for MVS is scheduled for 3Q97. Visigenic has an agreement with IBM that provides Visigenic with access to client-side CICS and MQSeries libraries. Visigenic is planning to release a comprehensive Legacy Integration Service by year-end.

2.3.5.2 OLE/COM or DCOM Integration

- Can the ORB be used by OLE clients?

  Response: The Visigenic COM\CORBA Bridge (VisiBridge) enables a COM client application to use a CORBA server. The COM client application may be one of these types:

- An OLE Automation controller like Visual Basic, Excel, or PowerBuilder
- A custom application written either in C++ or Java, using direct calls on COM interfaces.

Visigenic's COM\CORBA Bridge includes a Bridge Wizard that creates bridge objects automatically without any programming. The bridge objects that are created using the Bridge Wizard may be one of two types:

- An ActiveX control.
- A standalone DCOM server.

- Does the ORB use OLE servers?

  Response: Currently, VisiBroker does not have a mechanism for the ORB to access OLE servers. However, later in 1997, Visigenic's VisBridge will be bi-directIONAl.

- Can the ORB manage both OLE and CORBA repositories?

  Response: Browsing capabilities will be available to the OLE registry and will be tightly integrated with the next release of VisiBridge that will be bi-directIONAl.

- Do you have any plans regarding COM/CORBA and DCOM/CORBA gateways compliant with the OMG Interoperability specification?

  Response: The OMG spec maps from CORBA servers to COM clients and from COM servers to CORBA clients. The spec for both OLE Automation and direct COM interface; VisiBridge supports OLE Automation mapping from CORBA servers to COM clients and complies with OMG. The spec supports the widest number of client applications, including those written in VisualBasic, Java, and C++.

2.3.5.3 ORB Interoperability

Does the ORB have demonstrated interoperability with any other commercial ORB?

Response: Applications developed with VisiBroker are interoperable with other CORBA 2.0 compliant objects and with ActiveX/DCOM objects. VisiBroker provides a complete CORBA 2.0 Object Request Broker that uses IIOP as its native communications protocol. Unlike other ORBs, VisiBroker has no proprietary internal protocol and does not require a gateway or a translator protocol. Objects developed with VisiBroker are interoperable with other objects running on a CORBA-2.0 compliant ORB. It should be noted, however, that objects managed or accessed by an ORB other than VisiBroker would not be participants in VisiBroker's unique functionality, such as the high availability made possible by the Smart Agent architecture.

VisiBroker enables objects written in C++ or Java to be accessed by other objects written in other languages, including C++, Java, and Smalltalk.

Visigenic participated in the ObjectWorld East in 1996 and successfully interoperated with several other CORBA 2.0-compliant vendors. Today, you can find VisiBroker interoperating on the Internet at CORBAnet's web site with several other vendors. Since VisiBroker for Java was the first CORBA 2.0- compliant ORB it became the de facto standard for interoperability testing. Visigenic made this process easier by making available the Visigenic-developed test suites to other vendors for their use. Again, VisiBroker's mature implementation of IIOP was an important decision factor for both Netscape and Oracle in selecting VisiBroker to embed into their technologies.

2.3.6 OMG Services

Services provide essential capabilities that extend the basic operation of an ORB. The following set of services have been defined by the Object Management Group (OMG/TC) and have varying availability in commercial ORBs:

- Trading Service*

  Response: Visigenic plans to productize and GA the DTSC trader by the end of this calendar year (1997).

- Event Service*

  Response: Now, GA since December 1996 (C++ and Java).

- Lifecycle Service*

  Response: The OMG Lifecycle spec defines a standardized set of operations such as move, and copy that an object may support and some conventions regarding factories that an application can conform to. There is not, per se, anything that can be implemented as a service library that is linked into a server since the implementation of the Lifecycle interfaces is usually application-specific or at least involves a complex interaction of a number of services that is not easily encapsulated. VisiBroker does support the Lifecycle Service in the sense of providing client stubs for the interfaces and in the sense that our software follows the conventions.

- Security Service*

  Response: Visigenic is implementing security in two phases. The first phase is the secure ORB. VisiBroker for C++ 3.0 and VisiBroker for Java 3.0 have built-in support for SSL. The second phase is a complete implementation of the OMG security service. Visigenic is partnering with an organization that has exceptIONAl security expertise. Release 1.0 of the security service is slated for 4Q97.

- Transaction Service*

  Response: TPBroker, Visigenic's, and Hitachi's jointly developed implementation of OTS is currently in the pilot stage. Transaction semantics and guarantees are compliant to the OMG OTS specification. Visigenic is also providing XA support. Connectivity to TP monitors will be achieved either through IIOP interoperability or through custom modules in the Legacy Integration Service. Visigenic plans to provide Encina support through IIOP interoperability. In addition to TPBroker, Visigenic is creating its own version of OTS; it will be a "lightweight" version in full compliance with the spec. This will be available later in 1997.

- Relationship Service

  Response: Future.

- Licensing Service

  Response: Future.

- Externalization Service

  Response: Future.

- Naming Service*

  Response: Now, GA since December 1996 (C++ and Java).

- Persistence Service

  Response: No plan (OMG is in the process of obsoleting the current persistent object spec, we plan to track new OMG specifications as they emerge. Visigenic provides alternative persistence mechanisms today based on ODBC. VisiBroker features a DBAdapter that integrates persistent objects stored in OODBs such as ODI's ObjectStore.)

  Our engineering team is looking for customers/partners who would help us in "development partner/alpha customer" capacity, or at least review functIONAl specifications for object persistence functionality. They are looking for concrete customers that can dig down deeper to answer the detailed functIONAl questions that the engineering team will have.

- Concurrency Control Service

  Response: Planned FY98.

- Properties Service

  Response: Future.

- Object Query Service

  Response: Future.

- Time Service

  Response: Planned.

Please indicate which standard services are currently available or in progress. Please supply software version information (by platform if necessary) for available services, and anticipated delivery dates for expected services.

If proprietary services, comparable to some of the standardized or not-yet standardized services are provided, please indicate what your migration plans are towards standards compliance.

In addition, a number of services are in the process of standardization for CORBA 3.0. These include important services such as the asynchronous messaging service, as well as the portable object adapter. What plans do you have to support migration towards those and other upcoming standards?

2.3.7 Development Issues

ABCD 1.1 was designed and implemented using state-of-the art object-oriented tools and processes. Rational Rose is at the center of this process, supporting both iterative development and the Booch methodology. Rational Rose also generates C++, which is our core development language. ABCD's staff has acquired substantial OO expertise that it expects to leverage with a CORBA-based solution.

Questions in this section address various aspects of development facilitation.

2.3.7.1 Integration with a Development Environment

An integrated development environment such as Microsoft's Visual C++ or CenterLine's ObjectCenter can greatly assist the developer in building and debugging code.

- How difficult will it be to integrate the ORB into our existing development environment or other client-oriented development environments?

  Response: This really depends on the architecture of the "tools" in use at ABCD. This can be a relatively easy process.

- Is distributed object development using the ORB easily integrated with an open, integrated development environment?

  Response: VisiBroker for C++ includes an IDL-to-C++ stub and skeleton code generator.

  VisiBroker for Java includes an IDL-to-Java stub and skeleton code generator.

  VisiBroker for Java 3.0 will include the Caffeine technology jointly developed with Netscape. Caffeine enables Java programmers to write CORBA objects without having to use IDL.

  Third-party vendors are providing development tools that run on VisiBroker. For example, Borland has licensed VisiBroker for inclusion with JBuilder.

  Visigenic has plans for the development of distributed visual development tools.

- With a CASE tool such as Rational Rose?

  Response: Rational Rose simply generates CORBA 2.0-compliant IDL. This IDL can be run through VisiBroker's IDL compilers. Therefore, nothing special has to be done to use both Rational Rose and VisiBroker together.

- With an OOA/OOD methodology such as Booch?

  Response: Booch is just a methodology and not a product; there is no reason why ABCD cannot use this methodology with our product. (Rational Rose uses the Booch methodology.

- What hurdles do you anticipate for our staff as they migrate to an ORB-based solution?

  Response: With adequate training and mentoring with the experienced staff at ABCD, the transition to VisiBroker should be minimal.

2.3.7.2 Non-IDL-Based Development

Interface Definition Language (IDL) is not well-supported by development environments or current analysis or design tools. Such support enhances the usability of an ORB.

- Is the ORB integrated with an analysis/design tool?

  Response: Borland will integrate Visigenic's VisiBroker for Java object request broker (ORB) into JBuilder, Borland's next-generation development tool for building enterprise-wide multi-tiered client/server applications. Under the terms of the agreement, Borland will integrate VisiBroker for Java into JBuilder's client/server and enterprise versions of its software development tools. VisiBroker for Java will enable JBuilder developers to use the Internet Inter-ORB Protocol (IIOP) and link distributed application objects on local servers or across the Internet.

- Do any language development environments generate IDL compatible with this ORB?

  Response: VisiBroker will provide a set of utilities named "Caffeine". Caffeine's goal is to provide a *pure Java* solution for developers of IIOP-based distributed systems. One of the key requirements is that the Caffeine solution be able to provide transparent interoperability with ORB objects implemented in other languages such as C++. This functionality is available in VisiBroker 3.0.

  The components of Caffeine are

  - a code generator that takes Java interface files as input and produces IIOP compliant stubs and skeletons,
  - a VisiBroker for Java compiler that automatically produces CORBA IDL from Java code,
  - a default URL based name service, and
  - pass-by-value (C++ will also provide this functionality in VisiBroker 3.0).

2.3.7.3 Threads

Multithreading capability is listed both in this development section as well as in the Scalability section above. The reason for including this topic in both sections is that an architecture that requires significant concurrency, but that must be designed without being able to depend on a multithreading capability, is very different from one that assumes thread availability. Therefore, threading availability is both a development and scaling issue.

- Does the ORB support multithreading?

  Response: VisiBroker provides two sets of libraries a single-threaded library and another library that is thread-safe and re-entrant. If you use the multi-threaded version of the library, the VisiBroker core will automatically use threads for its internal processing, resulting in more efficient request management. For applications that never intend to use threads, the single-threaded library offers a set of interfaces similar to the multi-threaded library. This allows you to use the single-threaded library initially and then re-compile with the multi-threaded in the future with virtually no interface changes.

- Which threading models does the ORB support: thread-per-request, thread-per-session, thread-pool?

  Response: Object servers can choose between three thread policies; single threaded, thread-per-session, or thread-pooling. The single-threaded policy is automatically selected if you link your object server with the single-threaded library. The thread-per-session and thread pooling models differ in when threads are created and when they are released. You specify one of these thread policies by passing a BOA option to the server when it is started, or by hard-coding the desired thread policy in the BOA_init method invocation.

  Server Thread-per-Session Policy

  When your server selects the thread-per-session policy, a new worker thread will be allocated each time a client binds to one of the server's object implementations. A worker thread will be assigned to handle all the requests received from a particular client. When the client disconnects from the server, the worker thread is destroyed. This policy is more efficient that the single-threaded policy because it allows for more parallelism within the server.

  Server Thread Pooling Policy

  When your server uses the thread-pooling policy, it defines the maximum number of threads that can be allocated to handle client requests. A worker thread is assigned for each client request, but only for the duration of that particular request. When a request is completed, the worker thread that was assigned to that request is placed into a pool of available threads so that it may be reassigned to process future client requests. The overhead associated with the creation and destruction of worker threads is reduced because threads are reused rather than destroyed.

Client Application Threads

Client applications can use threads in several ways in relation to an object implementation. The client's main thread can obtain an object reference by invoking the bind method and pass the reference to each of the worker threads it creates. Each client worker thread can use the _clone method on an object reference passed by the main thread. Each client thread can issue its own bind request.

- If so, is the thread package platform independent, e.g., POSIX-compliant?

  Response: VisiBroker utilizes various thread packages based on the platform, for example: on Solaris, we use the UI thread API; on NT, we use its native threads; and on HPUX, DEC Alpha, AIX, and IRIX, we use the pthread libraries provided by the OS.

2.3.8 Management

Management of an ORB in a distributed high-volume, production environment that may include tens, hundreds, or thousands of machines, is a critical concern. Some of the issues to be addressed with respect to management include:

- Does the ORB support server management? (For example, can the ORB daemons be monitored and administrated from a single workstation or more advantageously, does the ORB use an operating system-independent method of administration?)

  Response: VisiManager, which will be released with VisiBroker 3.0, will have a tool that looks at what the Smart Agent knows about, and this is an object-centric view. Determining what servers are running would be indirect. AdditIONAlly, we will provide a list of all Activation Daemons running, and all Naming Factories (servers) that are running.

- Does the ORB support mechanisms for self-management?

  Response: The ORB, via the OSAgent process, maintains internal tables of object location. This lets the ORB transparently manage its object servers, with no intervention from the users.

- Can the ORB be integrated with a transaction monitor or will the product provide a CORBA 2.0-compliant transaction service that is integrated as part of the ORB?

Response: TPBroker, Visigenic's, and Hitachi's jointly developed implementation of OTS is currently in pilot stage. Transaction semantics and guarantees are compliant to the OMG OTS specification. Visigenic is also providing XA support. Connectivity to TP monitors will be achieved either through IIOP interoperability or through custom modules in the Legacy Integration Service. Visigenic plans to provide Encina support through IIOP interoperability. In addition to TPBroker, Visigenic is creating its own version of OTS; it will be a "lightweight" version in full compliance with the spec. This will be available later in 1997 and will be integrated with the ORB.

- Does the ORB provide or support a system management strategy (e.g., the Simple Network Protocol SNMP)?

  Response: SNMP support will be provided at a later date.

- Does the ORB provide or readily integrate with an alarm system for human escalation of exceptions?

  Response: Many customers built "alarm systems" for exceptions; now, especially with message interceptors, this has become even easier.

- Does the ORB provide or facilitate run-time instrumentation and reporting of performance metrics to facilitate tuning?

  Response: The VisiBroker ORB and the bind( ) method generated by the IDL compiler for each IDL interface can be selectively extended by developer-supplied C++ objects called interceptors. Writing an interceptor gives you the opportunity to supplement the operation of these components with your own code. You can, for example,

  - impose access controls on objects and methods, granting or denying access based on the requesting user's identity. Write log data for analysis of performance or usage patterns.
  - add to or alter the data passed between clients and servers, for example, add transaction information or encrypt a credit card number.
  - trace the routing of invocations from clients to (potentially forwarding) servers.
  - dynamically select the server that receives a client invocation, possibly choosing it based on load information.
  - provide an alternative location service that's invoked when the VisiBroker bind() method fails to find a server.
  - interceptors can be used with both the BOA and the Secure BOA.

2.3.9 Usability and Customizability

Usability and customizability are elements that can enhance or detract significantly from an ORB's utility.

2.3.9.1 Ease of Installation

- Does the product utilize a script that installs the ORB? Is the script reliable or does it fail? Can the script be used to remove the ORB?

  Response: Currently there is not script for installation; detailed instructions describe the simple process.

- Does the script assume some reasonable default locations for the ORB to be installed, consistent with the conventions of the platform?

  Response: Yes, default directories and paths are suggested but not enforced.

- Does the script update PATH variables and environment variables automatically?

  Response: Typically, system administrators will update applicable registries and class paths as necessary.

- Does the product install without damaging the current environment, e.g., overwriting other ORB's or related products?

  Response: No damaging effects should occur by installing VisiBroker. It can co-exist with other ORB and/or related products.

- Is the amount of time for installation consistent with similarly complex and similarly sized products on this platform?

  Response: Yes, the installation is a short process.

2.3.9.2 Ease of Configuration

- Is a script available for specifying connection time-outs, connection retries, diagnostic messages, etc.?

  Response: This is something more of a application-side issue. However, with OTS, many of these features will be inherent to the OTS service.

- Is configuration information available for inspection through a single mechanism, e.g., a command that prints all configuration settings for the ORB or a browser that supports viewing and editing?

Response: The osfind utility command reports on all VisiBroker-related objects and services running on a given network. By using osfind you can find out the number of osagent processes running on the network as well as the exact machine they are running on. The osfind command also reports on all VisiBroker objects that are active on the network. Use osfind to monitor the status of the network and locate stray objects during the debugging phase. VisiManager, which will be released with VisiBroker 3.0, will have a tool that looks at what the Smart Agent knows about, and this is an object-centric view. Determining what servers are running would be indirect. Additionally, we will provide a list of all Activation Daemons running, and all Naming Factories (servers) that are running.

- Are diagnostic messages regarding errors in configuration clear and definitive?

  Response: Yes, diagnostic messages are clear. In fact VisiBroker for Java 3.0 has a built-in debugging tool. Not all C++ compilers support exceptions through the try and catch statements, so the CORBA specification defines an Environment class for reflecting exceptions. VisiBroker uses the Environment class, along with a set of macros, to provide your applications with exception handling capabilities when try and catch are not supported.

2.3.9.3 Openness and Extensibility

- Can existing class libraries be incorporated into the ORB with minimal coding?

  Response: The question is under-specified. It depends on what sort of classes are in mind and what they do. VisiBroker 3.0 features a variety of extensibility capabilities running from interceptors to plug-in object adapters.

- Are constructs available in the product to allow extensions to be basic functionality? For example, constructs such as smart proxies, attribute caching, or metaclasses can be used to extend the ORB's basic functionality.

  Response: VisiBroker 3.0 will have smart proxies that allow caching of state local to the client, transparently to the client API. Also, we will provide interceptors API in order to manipulate the low-level IIOP buffers for infrastructure developers. Concerning metadata, this is already achieved with our support of the CORBA-compliant Interface Repository, which will also include some visual management tools.

- Are vendor-delivered CORBA Services available as shared libraries (.so, .dll) as well as executables so that they may be incorporated into user executables?

  Response: Yes. There are client stub libraries as well as server libraries (for the Naming and Events Services).

- How readily can the ORB event loop be integrated with standard client event loops?

Response: When your object implementation invokes the BOA::impl_is_ready method, an event loop is entered that waits for the arrival of requests from client applications. Your object implementation may also need to interact with another event-driven system. In a multi-threaded environment, you can solve this problem by simply using two threads, one thread that waits for VisiBroker events and another thread that services other events.

2.3.9.4 Portability of Customer-Developed Code

A particular ORB may require that client or server code observe conventions that are not easily portable between platforms, e.g., the exception-handling approach for Orbix Windows code is rather different than for Orbix UNIX code. Portability of code between ORB vendors also is problematic, server implementation code is very different between Orbix and ORB Plus; exception handling in ORB Plus C++ clients and servers does not follow C++ conventions and is very ORB specific. These questions attempt to characterize how portable an implementation is, between platforms and between vendors.

- Does the ORB support a consistent and portable exception-handling policy that is not dependent on the native compiling capability?

  Response: Yes.

- Is the client-side code easily portable to different platforms, using the same ORB?

  Response: Yes.

- Is the client-side code easily portable to a different ORB?

  Response: Yes, as long as no extensions are used, the ORBs would be portable.

  Can the server-side application code be re-targeted to a different vendor's ORB, given reasonable effort ?

  Response: Yes, as long as no extensions are used, the ORBs would be portable.

2.3.9.5 Documentation, Training, and Consulting Support

Solid documentation, training, and consulting support are absolutely essential for a large undertaking such as ABCD's.

- Is the documentation correct; e.g., are API's accurately defined or, if there are errors or obvious omissions in the documentation, what support is provided to readily supplement the published documentation?

  Response: Much of the API interfaces are documented. Please refer to the VisiBroker 3.0 documentation for an illustration of this. In fact, many of the API's are via IDL.

- Are the examples illustrative of fundamental concepts? Do they provide a graduated approach to understanding the ORB's features?

  Response: Yes, there are many examples of the ORB's functionality as well as code examples.

- Are there platform-specific examples, e.g., examples that illustrate ORB features that may need to be tailored to idiosyncratic elements of a platform?

  Response: There are no idiosyncratic elements pertaining to programming for different platforms.

- Is there adequate training on the ORB available?

  Response: Within the Visigenic Professional Services Organization, PSO is a fully trained and staffed teaching organization with over 20 staff years of experience. Training in VisiBroker for C++ and VisiBroker for Java are available and specialty courses can be arranged. These classes can be arranged at your location or at the Visigenic Training Center.

- Is expert consulting on the ORB available?

  Response: Within Visigenic ProfessIONAl Services Organization (PSO) we have staffed consultants that are certified to work with all Visigenic Software products. These consultants can perform both architectural and design work as well as staffing for projects. This can be done either on an on-site basis or through Visigenic's facilities.

  To assist you with your distributed application development issues, Visigenic has a staff of seasoned consultants available to help your project team at a critical point in the product development cycle. They deliver the technical expertise and objectivity needed to shorten the path to a successful IT solution.

  Visigenic consulting services are focused on distributed-object and data access middleware implementations. Our accomplished consultants have worked within these fields since the advent of the technologies themselves, achieving and maintaining the expertise necessary to guide clients to successful implementations of CORBA-compliant and ODBC/JDBC-based IT systems.
  Visigenic is the leading provider of embedded distributed object and data access technologies to the software industry. Such industry leaders as Netscape, Oracle, Novell, Borland, and Sybase have integrated our software into their products. As a result, our consultants can help you optimally leverage the functionality of the CORBA, ODBC, and JDBC technologies found in our partners' products.

Our consulting organization provides a wide variety of specialized services in the areas of enterprise application and database access consulting. Our consulting services focus on

Distributed Object and CORBA Application Architecture Development. A seasoned architect will lead or assist your project team in developing a distributed object architecture compatible with CORBA and DCOM concepts.

Strategy Assessment. We merge our in-depth knowledge of distributed object technologies and Visigenic products with your business and engineering goals to create a winning strategy.

Reference Application Design and Development. Our consultants can lead a development project to create or extend an application that provides a real-world example for other teams to reference.

Project Mentoring and Technology Transfer Consulting. Our VisiBroker product specialists can assist your project team members during implementation and deployment activities.

2.3.10 Business Issues

As noted earlier, factors relating to the overall health and strength of the company that sells an ORB may be of equal or greater importance in ORB evaluation than any technical concerns. Some of the issues relating to company evaluation are noted here.

2.3.10.1 Quality of Support*

- What are the levels of support offered: 24-hour availability hotline, daytime telephone support, e-mail only with some e-hour guaranteed response, bulletin board support, other?

  Response: Visigenic offers our customers a Premium Support Plan. It includes general terms and conditions, definition of case priorities, response times, closure of technical support cases, correction goals, escalation procedures, and maintenance releases. Please see appendix, *Premium Level Technical Support Services Agreement.*

- How responsive is the support organization?

  Response: 4-hour call-back time is guaranteed.

- How accurate or efficacious are support responses to customer questions?

Response: Visigenic supports a very complex product and hire extremely senior staff (often developers) to support it. We try to make our responses both accurate and efficacious. It does not always happen in a single call, because often one piece of information leads to the need for another. We do, however, try to get to the bottom of issues as soon as possible.

- Is there on-site consulting support available to aid the customer with product problems?

  Response: As mentioned above, Visigenic maintains a very senior staff that does provide various forms of on-site consulting support.

2.3.10.2 Viability of the Company

- Is the company well-capitalized?

  Response: Please refer to the enclosed Visigenic Prospectus.

- Does it have seasoned management staff?

  Response: Organization Chart and Senior Management Profiles.

Chief Executive Officer and Chairman

Roger Sippl

Roger Sippl is the Chief Executive Officer and Chairman of Visigenic Software, where he is the technology visionary responsible for long-term product development, strategic planning, and industry standards. Sippl has more than sixteen years of senior operations and chief executive experience working with technology companies.

Sippl is the founder of Informix Software, Inc. Under his direction, Informix pioneered SQL relational databases, 4GL application development tools, and OLTP database technology for the UNIX operating environment. Sippl served as the first chairman of the SQL Access Group, and served on the X/Open Board of Directors and the Independent Software Vendor Council (ISV). He also served on the UNIX InternatIONAl Executive Committee and is a founding board member of Uniforum, the first UNIX trade organization and show. In addition to his work at Visigenic, Sippl is co-founder and board member of The Vantive Corporation of Mountain View, California, a customer service automation software vendor.

Sippl holds a B.S. degree in Computer Science from the University of California at Berkeley.

President and Chief Operating Officer

Mark Hanson

Mark Hanson is the President and Chief Operating Officer of Visigenic Software, where he is responsible for all product development, strategic planning, sales, marketing, and operations. Prior to becoming President, Hanson was Executive Vice President of Sales and Marketing, with responsibility for worldwide sales, marketing, and support operations. Before joining Visigenic, Hanson held the positions of Vice President of Channel Sales at Sybase, and Vice President of InternatIONAl Sales at Gain Technology, prior to the Sybase acquisition. Hanson helped Sybase launch its high-end channels strategy, effectively building a channels organization from five employees to forty-three employees in eighteen months. In addition, the channels organization recruited over 200 new VARs and fifty systems integrators during this period.

Previous to Sybase, Hanson was at Macromedia, a leading supplier of PC multimedia software and services, where he held the position of Worldwide Vice President of Sales and Services. Before joining Macromedia, Hanson served as the Vice President of American Sales for Informix Software. During his seven years with Informix, Hanson played a key role in building the domestic sales organization from four people to over two hundred people. In addition, he helped build Informix's successful multi-channel distribution strategy, which included OEM, distributor, VAR, and direct sales organizations.

Hanson received a B.S. degree in Marketing from Santa Clara University, and his Masters in Finance from Golden Gate University in San Francisco.

Chief Financial Officer

Casey Eichler

Casey Eichler is the Chief Financial Officer for Visigenic, where he is responsible for finance, operations, information systems, human resources, and investor and shareholder relations.

Eichler has fourteen years of experience with software and publicly traded companies. Previously, he served as Executive Vice President and Chief Financial Officer for National Insurance Group, a publicly traded financial services and technology solution provider that employed over 500 employees and reported revenue of $36 million in 1995. Prior to that, he held the position of Executive Vice President and Chief Financial Officer at Mortgage Quality Management, where he helped to grow annual revenue from $500,000 to $23.5 million. Eichler also brought to Visigenic software experience through positions held with technology leaders Microsoft Corporation and NeXT Software, Inc.

Eichler earned his B.S. in Accounting from St. John's University. He is a Certified Public Accountant.

Vice President, Worldwide Sales

Scott Chalmers

Scott Chalmers is the Vice President of Worldwide Sales for Visigenic Software, where he is responsible for all sales activity, including direct sales, partner sales, OEM sales, and distributor sales.

Chalmers has twenty-five years of experience in senior sales roles working at technology companies. Prior to joining Visigenic, Chalmers spent six years at Informix Software, where he held positions as District Manager, Western Regional Manager, and Western Regional Vice President before assuming the duties of Vice President of U.S. Sales. He has also served as Area Director at AT&T Information Systems, worked as an independent consultant, and held various senior sales positions in a thirteen-year engagement with IBM.

Chalmers holds an M.B.A. degree from the University of California at Los Angeles. He received a B.A. degree from Oklahoma University.

Vice President, Marketing

Bob Macdonald

Bob Macdonald is Vice President of Marketing for Visigenic, where he is responsible for the strategy, planning, and execution of marketing programs for data access and distributed object products.

Macdonald brings to Visigenic thirteen years of strong marketing and sales experience in the enterprise and Internet software markets. Previously, he was Vice President of Corporate Marketing at Informix Software, where he played a central role in the planning of all product launches and acted as the primary spokesman for the company. Macdonald held other positions at Informix, including Executive Director of Strategic Planning and National Sales Manager.

Macdonald has diverse experience outside of high technology as well, including special effects production for a number of major Hollywood motion pictures, as well as working as the host of a nationally syndicated current affairs radio program in the U.S.
Macdonald earned his B.A. in Business Administration from Principia College in Saint Louis, Missouri.

Chief Technology Officer

Jens Christensen

Jens Christensen, Ph.D., is the Chief Technology Officer for Visigenic, where he is responsible for defining product strategies and technology directions.

Christensen was co-founder and CEO of PostModern Computing, which merged with Visigenic in 1996. Before his tenure with PostModern, Christensen held engineering, consulting, and management positions with Columbia University, FMC Corporation, and Teknekron Corporation.

Christensen holds a Ph.D. in Computer Science from Stanford University.

Vice President, Engineering

Richard Kelman

Richard Kelman is the Vice President of Engineering for Visigenic, where he manages development, product management and quality assurance of the company's distributed object and data access technologies.

Prior to joining Visigenic, Kelman was Vice President of Client Services Development for Tesseract Corporation, a leading provider of human resources management systems. He has also served as Director of Systems Development for Dow Jones and Vice President of Market Applications Development at Telerate Systems. Kelman held a variety of positions at UNI VAC Corporation, where he worked for 18 years.

Kelman studied Mathematics at Edinburgh University.

Vice President, Corporate Development

Richard L. Gerould

Richard Gerould is the Vice President of Corporate Development for Visigenic, where he is responsible for creating and maintaining relationships with strategic customers and suppliers.

Prior to joining the company, Gerould founded and served as President of Configurex, Inc., a developer of object-oriented rapid development systems for distributed, cross-platform applications. Gerould has practiced as an attorney, manager, and engineer for other prominent Bay Area Firms. Before founding Configurex, Gerould was the Vice President of Marketing Operations at Micro Focus, Inc., a company with annual revenues exceeding $100 million. Gerould also served as the Vice President of Corporate Services during his tenure at Micro Focus.

Gerould holds a B.A. and M.S.E.E. degree in Computer Science from the University of California at Berkeley. He also holds a J.D. degree from Hastings College of Law in San Francisco.

Vice President, ProfessIONAl Services

Robert Perreault

Bob Perreault is Vice President of Professional Services for Visigenic, where he is responsible for the company's consulting operations. He previously served as the company's Vice President of Research and Development.

Perreault has fifteen years of expertise in database research and development. Prior to joining Visigenic, he held the position of Vice President of Client/Server Technology at Compuware Corporation. He also served as Vice President of Database and Connectivity Products and Vice President of U.S. Engineering at Uniface Corporation, which merged with Compuware in 1994. At Uniface, Perreault helped deliver transparent access to a variety of sources through a series of network and database drivers. Previous to Uniface, Perreault co-founded and served as President of Data Accessibility Solutions, Inc., was Vice President and co-founder of RIAL, Inc., and held the position of Vice President of Engineering at Interactive Development Environments. He also worked in research and development at Hewlett-Packard for ten years in the Information Technology Group and the Corporate Administrative Systems division.

Perreault holds an M.S. degree in Computer Science from Stanford University, an M.B.A from the University of Michigan, and a B.A. in Economics and a B.S. in Math Sciences from Stanford University.

Résumés of Key Personnel Who are involved in the Product Research and Development of the Product.

Director, Object Technologies **Neguine Navab**

Neguine Navab is responsible for development of Visigenic's VisiBroker object request broker (ORB) products and related object services such as naming and events.

Navab was Vice President of Product Development for PostModern Computing, which merged with Visigenic in 1996. Prior to Visigenic, she held the role of Manager of the Distributed Object Group (NEO) at SunSoft, where she oversaw development of the communication core of Sun's ORB implementation. Navab also worked as a developer and project lead in SunSoft's Solaris Kernel Operating System group. Prior to joining Sun Microsystems, she was with Oracle Corporation.

Navab holds an MS in Computer Science from Stanford University and BA degrees in Computer Science and Math from the University of Oregon.

Director, Engineering **Farid Khoujinian**

Farid Khoujinian oversees the development of Visigenic's object services technology. His Strategic Object Services group develops modular, system-level services that complement the functionality of VisiBroker ORB products.

Khoujinian has led technical engineering and consulting engagements in all facets of the information technology industry. Prior to joining Visigenic, he was the Director of Advanced Solutions within KPMG's Strategic Services Consulting group, where he focused on providing solutions to clients with highly challenging development projects. He also served as the lead architect for the implementation of Sybase's next-generation database technology and developed the company's distributed objects and connectivity strategy.

Khoujinian earned a Ph.D. and M.S. in Electrical Engineering from Columbia University. He received a B.S. in Applied Mathematics from Harvard University.

Senior Product Manager **Geoffrey Lewis**

Geoffrey Lewis is Visigenic's Senior Product Manager for object products. His responsibility is managing the development and delivery of distributed object technology, including the VisiBroker object request brokers, services, and development tools.

Lewis has 25 years of software industry experience. Prior to joining Visigenic, he held a number of positions in the SunSoft division of Sun Microsystems. His roles there included Manager of Strategic Alliances for SunSoft Object Products, Manager of Strategic Endorsements, Engineering Manager, and Project Leader. He has also held positions as Group Manager of Database Products at Apollo Computer; Principal Software Engineer at Lotus Development; Manager of Application Systems and Languages at Computer Consoles; and System Architect at I.P. Sharp Associates. Lewis began his career as a lecturer for the Department of Computing and Control for the Imperial College in London.

Lewis has received commendations from the Object Management Group, including a Distinguished Service Award in 1994 and OMG's first Fellows Honor in 1997.
Lewis earned an MS in Computer Science from the University of London and a Bachelor of Science degree in Theoretical Mechanics from the University of Nottingham.

Résumés of Key Personnel Involved in the Professional Services Organization

Director, Advanced Business Solutions **Kevin Riley**

Kevin Riley's responsibilities include providing presales consultation on use and integration of the company's distributed object and data access technologies, as well as ongoing account management duties.

Riley has more than 20 years of experience as a computer professional. Prior to joining Visigenic, Riley held a number of positions at New York Life, one of the largest financial institutions in the world. Holding the position of Chief Architect, he set the company's overall application scheme, system strategy, and technology direction. He also served as Systems Director for the Corporate Technology Group; Systems Director for MIS Network Support; Systems Manager for Designer Database Development and Support; Systems Manager for CICS; IDMS Support and Application; and Systems Programmer.

Director, Consulting Services **Dale Lampson**

Please see Appendix A1, Résumés

Principal Consultant **Carlos Muchiutti**

Please see Appendix A1, Résumés

Principal Consultant **Rama Penumarti**

Please see Appendix A1, Résumés

Principal Consultant **Andreas Vogel**

Please see Appendix A1, Résumés

- Does it have sufficient technical staff to evolve and support the product?

  Response: Visigenic is committed to continuing to evaluate its Customer Support capabilities to ensure they continue to meet the needs of our customers' dynamic environments. To that end, Visigenic has implemented several new programs in the last six months which allow us to more effectively respond to the different classes of customers we have. Visigenic is currently in the process of an independent study of its customer base to develop additional capabilities for the coming year.

- Is the company based in a stable government and economy?

  Response: Visigenic's headquarters in San Mateo, California. This is in the heart of the Silicon Valley, which is one of the richest (if not the richest) concentration of technical companies and expertise.

- How long has the company been in business and actively working with this technology?

2.3.10.3 Alacrity of the Company

How effectively can the company respond to changing needs of the industry or of its key customers?

2.3.10.4 Dedication of the Company to the Product

- Is the company dedicated to this product as part of its long-term strategic direction?

Response:

Visigenic Strategy

Visigenic's strategy is to become the premier provider of software tools which enable developers and IT professionals to develop, deploy and manage distributed business applications for Internet, Intranet and enterprise computing environments. The Company's strategy incorporates the following key elements:

Support and Enhance Open Industry Standards. The Company's products are based on existing and emerging industry standards for heterogeneous database access and distributed object technology. The Company actively participates in standards-setting organizations including X/Open and the Object Management Group. The Company intends to contribute to the expansion of existing standards and the development of future standards created by these and other standards-setting organizations. For example, the Company was instrumental in accelerating ODBC's acceptance as a standard for heterogeneous database access by licensing certain ODBC enabling technology from Microsoft and providing the VisiODBC SDK on Macintosh, OS/2

and many UNIX platforms. In addition, the Company has developed the first commercial implementation of IIOP in its ORB products. The Company intends to continue promoting acceptance of IIOP as the de facto standard for distributed object messaging for the Internet and Intranets.

Leverage Strategic Partners. The Company intends to continue to establish close relationships with leading technology companies through technology licensing, joint development, strategic investments, and distribution and marketing arrangements to promote the widespread acceptance and distribution of Visigenic products. Key partnerships include the following:

- ◊ *Cisco* is a strategic investor in the Company and has entered into a license agreement to incorporate the Company's database access products in Cisco's network management product line.

- ◊ *Hitachi* has entered into a joint-development agreement with the Company for the development of a transaction-enabled ORB based on the VisiBroker for C++ and VisiBroker for Java products and is a worldwide distributor of the VisiBroker product line.

- ◊ *Microsoft* has entered into a series of agreements to incorporate certain of the Company's VisiODBC products with certain Microsoft products and to license to the Company the Microsoft ODBC SDK and test suites for non-Microsoft operating systems.

- ◊ *Netscape* is a strategic investor in the Company and has entered into a license agreement to incorporate VisiBroker for Java into Netscape's client products, Navigator 4.0 and Communicator, and VisiBroker for C++ into Netscape's server products, Enterprise 3.0 and FastTrack 3.0.

- ◊ *Oracle* has entered into a license agreement with the Company to incorporate the Company's VisiODBC products with a number of Oracle's Transparent Gateway products.

- ◊ *Platinum technology* is a strategic investor in the Company and has entered into an agreement to incorporate the Company's VisiODBC, VisiChannel and VisiBroker products with certain Platinum technology products.

Provide a Broad Suite of Integrated Products. The Company offers a suite of software tools for database access and distributed objects. The Company intends to develop new products, enhance its current products and continue to integrate its database access technology with its distributed object technology to address the requirements of the development, deployment and management of business applications.

Leverage Product Sales by Providing Professional Services. To address the growing demand for expertise in IIOP, CORBA and the development of distributed object applications, the Company employs a staff of professional consultants and trainers who are experienced in database access and distributed object technologies. The Company intends to continue to develop and grow its professional service organization as a key differentiating component of its sales strategy.

Maintain Technology Leadership. The Company is committed to maintaining its technological leadership through internal product development efforts and, if appropriate opportunities present themselves, through acquisitions of technologies, products and companies to address the specific requirements of distributed applications in the areas of database access, distributed objects and the monitoring of distributed environments. The Company has invested and will continue to invest in technology so that it can react and adapt to changing technological trends and market needs.

Exploit and Develop Internet/Intranet Market Opportunities. The Company believes that the emergence of the Internet and Intranets will significantly increase the market for database access and distributed object connectivity software. Visigenic intends to leverage its products and expertise in heterogeneous database access and distributed objects to exploit the market opportunity for distributed applications for the Internet and Intranets.

Expand Distribution Channels Worldwide. To achieve broad distribution of its database access and distributed object software, the Company believes it must continue to build multiple distribution channels worldwide. The Company is expanding its direct sales and telesales forces as well as broadening its indirect channels of distribution, including VARs, ISVs, systems integrators ("SIs"), Internet sales and international distributors. The Company's international distribution strategy is to penetrate key international markets by seeking additional VARs, ISVs and regIONAl distributors and by further developing its existing relationships with these customers.

- Is this product an acquired product that may be retooled or dropped in favor of another company product?

  Response: No, in fact Visigenic's focus is on the ORB. We have recently re-engineered our ODBC server product line (VisiChannel) to utilize the ORB as its underlying architecture.

- Is the company committed to the product both financially and technically and further, do they have the resources to see through the commitment?

  Response: Visigenic is total committed. The following was announced last month:

SAN MATEO, CA - May 5, 1997 - Visigenic Software, Inc. (NASDAQ: VSGN), the leader in distributed object technology for Java and the Web, was named the fourth fastest growing public company in Silicon Valley by The San JoseBusiness Journal. Visigenic was named number four based on the company's outstanding average revenue growth of 301 percent over the past three years, ending March 31, 1996. For the past twelve months ending March 31, 1997, Visigenic's revenues have increased 205 percent to $17.0 million from $5.6 million the prior year.

The Business Journal included companies based in Silicon Valley that were at least three years old at the end of the 1996 fiscal year. The organizations were ranked by William O'Neill & Co., Inc. using a method called "least squared." This process takes into account both growth spurts and revenue drops, transforming them into a smooth growth curve.

- Does the company play a major role in the OMG and the CORBA standard (e.g., sit on the board of directors)?

  Response: Visigenic is a major player in the OMG. Jeff Mischkinsky is Visigenic's OMG representative, this is Jeff's full time job. Jeff has been on the Architecture Board since it was formed. He is the Chair of the Enhanced Portability Revision Task Force and IDL/Java Language Revision Task Force. (AB members are not allowed to Chair regular Task Forces). Jeff has been active in the OMG since 1991. He served on the OMG Board of Directors (representing Unify and then Sybase) for several years. Jeff was on the original 90-day team that delivered the original CORBA 1 specification. Jeff was the primary drafter of the CORBA Component and CORBA Scripting RFP's which are being submitted as new RFP's to be issued.

  Visigenic is an active participant in many RFP's. Most recently our submissions for the Java Language Mapping and the CORBA SSL/Security Service were adopted by the OMG. We are currently submitting to the following RFP's:

  Objects By Value, Messaging, Notification, and Com/CORBA Part B. We will be submitting to the reverse Java To IDL Mapping RFP

2.3.11 Pricing Issues

Licensing costs are another significant factor affecting ABCD's ability to deliver cost-effective solutions to the TBUs.

Please indicate licensing cost information relative to ABCD's development and deployment requirements for:

- SOC runtime licenses
- Development licenses
- BUS runtime licenses

- Client runtime licenses (WebServer as well as compiled client)

Response: Pricing issues will be discussed at a later time

2.3.12 Third-Party Products

We expect that CORBA-enabled COTS products will soon emerge on the market. Please list any existing products that you are aware of, whether currently available or planned. Also please indicate if you have any strategic arrangements with other vendors that will impact the rate at which products integrated with your ORB are likely to enter the market.

We are particularly interested in products pertaining to Virtual Community services, Delivery Services (Fax, Off-line Print), Electronic Commerce, and Web Server Frameworks.

2.3.13 Case Studies

Please provide case study material that will support the viability of your product for high-volume, enterprise-strength applications.

Please include information pertaining to:

- Time-to-market
- Number of services, servers, sites, and clients
- Reliability, MTBF
- Response time
- Maintenance costs

References will be provided upon request

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1998	Contractor Report

4. TITLE AND SUBTITLE
Aviation System Analysis Capability Executive Assistant Design

5. FUNDING NUMBERS
C NAS2-14361
Task 97-01

WU 538-08-11-01

6. AUTHOR(S)
Eileen Roberts, James A. Villani, Mohammed Osman, David Godso, Brent King, and Michael Ricciardi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Logistics Management Institute
2000 Corporate Ridge
McLean, VA 22102-7805

8. PERFORMING ORGANIZATION REPORT NUMBER
NS701S1

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA/CR-1998-207679

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Robert E. Yackovetsky
Final Report

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 01

Distribution: Nonstandard

Availibility: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

In this technical document, we describe the design developed for the Aviation System Analysis Capability (ASAC) Executive Assistant (EA) Proof of Concept (POC). We describe the genesis and role of the ASAC system, discuss the objectives of the ASAC system and provide an overview of components and models within the ASAC system, and describe the design process and the results of the ASAC EA POC system design. We also describe the evaluation process and results for applicable COTS software. The document has six chapters, a bibliography, three appendices and one attachment.

14. SUBJECT TERMS
ASAC, NASA, Design, Executive Assistant

15. NUMBER OF PAGES
211

16. PRICE CODE
A10

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited