CAGE Code 81205

Real-time CORBA Trade Study

Volume 1

DOCUMENT NUMBER:	RELEASE/REVISION:	RELEASE/REVISION DATE:
D204-31159-1	**ORIG**	

CONTENT OWNER:

Phantom Works Advanced Information Systems (9-5430)

All future revisions to this document shall be approved by the content owner prior to release.

Document Information

Document Type	Original Release Date	Contract Number (if required)
Formal	January 10, 2000	F34601-96-C-0720 98-ESC-13
Preparing Organization (if different from owning organization)		**Hardware and Software Used** IBM PC Microsoft Word 97
Location of Software Files (optional)		
Boeing Web URL (optional) http://diicoe.web.boeing.com		
Notes and Limitations (optional) ~~Not approved for foreign disclosure.~~ Amended for public release by ESC/PAM 31 January 2000		

Signatures for original release

AUTHORS:	H. Rebecca Callison and Daniel G. Butler	9-5430	/s/ 13 Dec 1999
	Sign and type: First Name MI Last Name	Org. Number	Date
APPROVAL:	Marilynn B. Goo	9-5430	/s/ 13 Dec 1999
	Sign and type: First Name MI Last Name	Org. Number	Date
DOCUMENT RELEASE:	Sheilah J. Carnahan	G-8214	/s/ Jan. 10, 2000
		Org. Number	Date

Table of Contents

List of Figures

List of Tables

Abstract

In 1999 the Defense Information Infrastructure Common Operating Environment (DII COE) Real-time Integrated Product Team (RT IPT), in conjunction with the RT Technical Working Group (TWG), conducted a study of Object Request Brokers (ORBs) of interest to the Department of Defense (DoD) real-time community. The products that the IPT and TWG chose for evaluation included (1) HARDPack, from Lockheed Martin Federal Systems, (2) ORBexpress RT, from Objective Interface Systems, and (3) The ACE ORB (TAO), from Washington University and Object Computing, Inc.

In performing the evaluation, Boeing, the IPT contractor, tested each of the three products using both Sun/Solaris and PowerPC/LynxOS platforms. In addition, the IPT conducted a vendor survey of technical and non-technical product characteristics and a user survey on topics ranging from product features to product support. Based on our findings, the IPT and TWG concluded that both ORBexpress and TAO deserve further consideration by specific applications seeking an ORB. Details about all aspects of the evaluation are presented in this document.

Limitations

The testing performed as part of this evaluation is only a subset of what should be done in a comprehensive study. We assessed general purpose aspects of ORB behavior but did not test real-time characteristics such as bounded operation times, multi-threading, priority inversion, and resource control.

1. Introduction

1.1 Scope

This document describes an evaluation of real-time (RT) Common Object Request Broker Architecture (CORBA) products undertaken between June and October, 1999. The work was performed under contract F34601-06-C-0720, Engineering Service Task (EST) 98-ESC-13, "Common Operating Environment (COE) for Real-time Systems". Results included in this report are limited to the characteristics of the products and vendors at the time the tests were performed or surveys conducted.

1.2 Objective of Study

The objective of the study is to identify viable RT CORBA products for recommendation into the DII COE Infrastructure. The study addresses both technical and non-technical aspects of the CORBA products. While individual programs may choose to use it as an initial filter, it is not intended to select a product for use in specific systems.

1.3 Background

The US Department of Defense, through its Defense Information Systems Agency (DISA), established the Defense Information Infrastructure Common Operating Environment (DII COE) to increase interoperability and facilitate reuse across defense C4 systems. DII COE history and requirements are described in the DII COE Integration and Runtime Specification (I&RTS), reference 1. In 1997, at the request of the Air Force, Army, Navy and Marine Corps, DISA chartered a DII COE Real-time Technical Working Group (TWG) to develop a set of common requirements and recommendations for potential products to provide real-time capabilities to the DII COE. In late 1997, the Air Force designated the Airborne Warning and Control System (AWACS) Program Office as Executive Agent for DII COE real-time extensions. The Executive Agent, currently Lt. Col. Lucie Robillard, leads the DII COE Real-time Integrated Product Team (DII COE RT IPT). The IPT evaluates, nominates and integrates new segments into the DII COE supporting the real time requirements established by the TWG. Because their missions are so closely related, the Real-time TWG and IPT are in continuous coordination, conduct joint meetings and share data[1].

The rapid evolution of CORBA for RT and the increasing availability of RT CORBA products in the marketplace prompted the IPT to fund The Boeing Company to perform a study of products available during the period from June through October, 1999. Preliminary surveys identified three key products of interest to the DII COE RT community:

[1] Reference 2, “Extending the DII COE for Real-Time.”

- HARDPack, the CORBA product offered by Lockheed Martin Federal Systems (LMFS), Owego, NY, and currently baselined on the USAF Airborne Warning and Control System (AWACS) and Regional / Sector Air Operations Center (R/SAOC) programs;
- ORB*express* RT, a product of Objective Interface Systems (OIS), Reston, VA; and
- The ACE ORB (TAO), an open source product of Washington University, St. Louis, MO, that is commercially supported by Object Computing, Inc., also in St. Louis.

At the time the study was conducted, all three products were available commercially. In a telecon with Boeing on November 12, 1999, LMFS stated that it will focus its CORBA expertise on support of existing HARDPack customers and pursuit of CORBA application/system integration opportunities. LMFS will refer new ORB opportunities to OIS and stated that the features of HARDPack will be integrated to provide add-ons to ORB*express*. OIS confirmed that they had reached an agreement with LMFS. OIS intends to examine the customer demand for services such as Interface Repository and Fault Tolerance included in HARDPack but not currently in ORB*express.* When sufficient market demand exists, OIS expects to offer these services as options available for its ORB*express* product line. HARDPack results are reported in this document but should be considered with these developments in mind.

As stated above, the goal of the study is not necessarily to recommend a specific ORB for use. We seek to identify any and all products that appear to be suitable for use in the real-time extensions to the DII COE. Technical assessments address three areas: standards compliance, basic performance, and interoperability with other ORBs. Other assessments deal with the usability of the product plus a cursory examination of the business viability of each vendor and product.

At best, the results of the study provide only preliminary insight into selection of a real-time ORB for use in a specific program. Individual selections must be made based on specific requirements of the individual program. Requirements discriminators may exist in areas such as physical environment, type and stringency of timing requirements, language and compiler choices, software architecture, and communications strategies. Individual programs considering the use of a real-time ORB should perform system-specific analyses and testing before making implementation choices.

1.4 Organization of Document

We describe the overall organization of the study in Section 2 and then present data and results from the various aspects of the study in separate sections of the report as follows: technical assessments appear in Section 3, assessment of vendor support in Section 4, and business related factors (vendor viability, cost) in Section 5. Section 6 summarizes conclusions and presents recommendations.

Comprehensive performance measurements, vendor survey responses, and user survey responses obtained through the study are not included in the report. The full results may be obtained by government programs with a valid interest in the data by contacting:

Lt Col Lucie M.J. Robillard, USAF Lucie.Robillard@hanscom.af.mil.
AF Executive Agent for Real-time DII COE
HQ ESC/AWW
3 Eglin Street
Hanscom AFB, MA 01731

2. Overview of Study

The scope of the study was to evaluate three products: HARDPack from Lockheed Martin Federal Systems, ORB*express* from Objective Interface Systems, and The ACE ORB (TAO) from Washington University in St. Louis with product support from Object Computing, Inc. The three basic elements of the study included (a) vendor survey, (b) independent assessments, and (c) user survey.

2.1 Vendor Survey

The initial vendor surveys were distributed by email July 21, 1999. Responses were received, reviewed, and analyzed between August 6 and August 19, 1999.

As the first step in the vendor survey, the RT IPT developed three questionnaires. The first, Technical Questionnaire, focused on the degree to which products satisfy the set of requirements established for real-time distributed computing infrastructure by the RT TWG. The second, Non-Technical Issues Questionnaire, dealt with questions of vendor and product viability, strategies for maintenance and evolution, and compliance with CORBA standards. Finally, the third, RT CORBA 1.0 Conformance Statement Questionnaire, was included to assist in establishing the degree to which the recently released OMG real-time standard is supported. The questionnaires are included as Appendix A.

Responses from all three vendors were received and distributed to a set of "expert reviewers" from organizations currently using or studying RT CORBA products. These reviewers represented organizations including the Joint Tactical Terminal Program Office, the Naval Surface Warfare Center, and MITRE – Bedford, and Boeing programs beyond the RT DII COE IPT. Boeing combined their analyses with its own and presented the results at the August 24, 1999 RT DII COE IPT meeting. After the meeting, follow-up questions to each vendor were prepared. Vendors were interviewed by telephone and/or responded to additional questions electronically.

2.2 Independent Analyses

Independent analyses included three types of tests run in the RT DII COE Integration Facility: Basic Data Integrity, Basic IDL Performance Benchmarks, and Interoperability Tests.

2.2.1 Basic Data Integrity (BDI) Test

The purpose of the basic data integrity test was to verify correct transfer of data when communicating via a single ORB in a heterogeneous environment. Test cases covered communication between applications running on different platforms and written in different languages.

2.2.2 Basic IDL Benchmark Tests

The purpose of the Basic IDL benchmarks was to measure the time required to transfer data of various sizes and types between a single client and a single server. These tests are basic in that they

measure only simple performance of isolated ORB operations. They do not address real-time behaviors.

The test measures performance for two invocation options: (1) One-Way (OW) Method: Client continues execution after invoking the operation without waiting for the server to complete the request and (2) Call & Return (CR) Method: Client execution is suspended until the server completes the request. For each specific interface under test, an operator-selectable number of operation invocations are executed. The individual execution time of each client operation is measured and recorded as is the latency between client invocation and activation of a servant application by the server. The test client reports average, minimum, and maximum operation execution times plus standard deviations calculated from the sample data. The same statistics are reported on the server side for latencies.

2.2.3 Interoperability Tests

The interoperability tests were performed to evaluate the capability of the RT ORBs under test to interoperate with each other and with a non-RT ORB. The RT ORBs resided in the same host and in different hosts. The RT ORBs under test were ORB*express* and TAO executing on PowerPC/LynxOS and Sun/Solaris[2] platforms. The non-RT ORB used was Orbix 3.0 running on a Sun/Solaris platform. Both the Ada95 and C++ versions of ORB*express* were exercised. HARDPack was not used because the test team was unable to execute tests using its IIOP capabilities. For the purposes of this test, Basic ORB Interoperability was defined as the ability of multiple ORBs to conduct reliable two-way communications. Note that this is *not* equivalent to "Interoperability Compliance" as defined in CORBA specifications. The non-real-time ORB parts of the test were run by the Boeing Phantom Works Distributed Software Technology organization using its facility in Bellevue, WA.

2.3 User Survey

In September 1999, the IPT prepared a user questionnaire designed to determine which RT CORBA features are most needed. Another objective of the questionnaire was to get a feel for user opinions about product quality and product support. The questionnaire was distributed via email to users identified in the vendor survey and to other individuals and organizations known to be using at least one of the three RT CORBA products being evaluated. Survey results are discussed in section 4.1. The questionnaire is provided as Appendix B.

[2] Solaris is an operating environment that includes the SunOS operating system. Both terms are used in this document.

3. Technical Capability Assessments

Technical capabilities of the subject ORBs were assessed in several ways. In section 3.1 we report the results of a vendor self-assessment with respect to the distributed computing requirements established by the DII COE RT TWG as of July 1, 1999. Section 3.2 provides an overview of connection and concurrency management of each of the ORBs, and Section 3.3 summarizes information on product availability and portability. The results of ORB tests and analyses conducted by the RT DII COE IPT are presented in Section 3.4. ORB technology, particularly for real-time, is evolving rapidly, so analyses of particular products quickly become outdated. The following products were the most current available from the ORB vendors at the time benchmarks and analyses were conducted, and, unless otherwise noted, specific evaluation results are valid only with respect to these particular products:

ORB Vendor	Specific Product Under Evaluation
Lockheed Martin Federal Systems Owego, NY	C++: HARDPack 1.2.1 (patch release)
	Ada95: HARDPack 1.2.1
Objective Interface Systems Reston, VA	C++, Solaris: ORB*express* GT 2.1.2
	C++, LynxOS: ORB*express* GT 2.1.4B-BETA
	Ada95, Solaris: ORB*express* RT 2.0.1
Washington University / Object Computing, Inc. St. Louis, MO	C++ for Solaris and LynxOS: TAO 1.0.1 (with ACE 5.0.1)

ORB*express* comes in a variety of product flavors with respect to capability and performance characteristics: (1) ST, a standard ORB; (2) GT, a configurable, performance-oriented variant intended for use in embedded systems without real-time requirements; and (3) RT, a variant with GT configurability and performance plus awareness of priority that enables real-time (predictable) behaviors. Further, OIS is in the process of implementing a more configurable, higher performance "next generation" architecture. At the time of the IPT's hands-on evaluation, the new architecture was available in GT version for C++ on both Solaris and LynxOS (beta only) with RT product releases planned for 4th quarter of calendar 1999. The most recent Ada product available was RT 2.0.1, a version based on the older architecture. In the vendor self-assessment described in 3.1, OIS was instructed to respond regarding current or planned capabilities for its RT products only. These responses for ORB*express* RT should not be construed as automatically applying to other members of the OIS ORB family.

TAO's portability is achieved by using framework components from the Adaptive Communications Environment (ACE), so ACE is a necessary part of the TAO execution environment. ACE is an open-source object-oriented (OO) framework developed at Washington University. ACE provides a set of reusable C++ wrapper facades and framework components that perform common communication software tasks across a range of OS platforms. ACE must be loaded into the development environment and the ACE components required by TAO must be available for linking into executables for the target environment.

3.1 Vendor Self-Assessment

We used both vendor self-assessments and independent user assessments to determine the extent to which each RT CORBA product satisfies the requirements defined by the RT TWG.

3.1.1 Vendor Commitment to Support RT CORBA 1.0 Standard

The first requirement of the RT TWG for RT-capable CORBA products is conformance with the RT CORBA 1.0 specification (reference 5). Washington University projects availability of a RT CORBA conformant TAO product at the end of 1999. OIS plans its release of a RT CORBA-conformant product in the spring of 2000. The issue of future releases of HARDPack has ceased to be of interest to the IPT since HARDPack is no longer being offered to new customers.

3.1.2 Availability of CORBA Standard Capabilities

Table 3-1 summarizes the standard capabilities and services of the three CORBA products as reported by the vendors. Discussions related to particular responses follow.

Table 3-1. Standard Capabilities and Services

Capability/Service	HARDPack	ORB*express*	TAO	Comments
IIOP	Non-native. Solaris only (others planned)	Native (default) protocol of ORB	Native (default) protocol of ORB	
IDL Compiler	Yes	Yes	Yes	
IDL Bindings				
ANSI C	Yes	No	No	
C++	Yes	Yes	Yes	
Ada 95	Yes	Yes	No	
Java	On demand	Future?	Future?	
Ada 83	Yes	No	No	
Static Invocation Interface	Yes	Yes	Yes	
Naming Service	Yes	Yes	Yes[1]	[1] Naming services support both persistent & non-persistent objects
Event Notification Services	Yes	Future	Yes[2]	[2] CORBA events
Time Service	Proposed enhancement to spec	Future	Yes[3]	[3] CORBA time
Implementation Repository	No	Yes	Yes	
Interface Repository	Yes[4]	Future	Future	[4] C, C++
Trader Service	No	No	Yes	
Dynamic Invocation Interface	Yes[5]	Yes[6]	Yes	[5] C, C++ [6] API is not standard-compliant for performance reasons
Transaction Service	No	No	Yes[7]	[7] Supplied by 3rd party: www.arjuna.com/products/OTS
Asynchronous Messaging	Includes subset of asynchronous messaging APIs	Partial – Yes Full – Future	Partial – supports callback model	

IIOP. The HARDPack ORB uses a proprietary message protocol as its native protocol, while ORB*express* and TAO ORBs use IIOP as their native (or default) protocols. HARDPack requires that the user identify objects that will interact using IIOP, rather than the proprietary protocol, at the time the object interface is initialized. IIOP-based requests are then routed to a HARDPack-specific IIOP server. The server then relays each message to the destination object under IIOP. In HARDPack release 1.2.1, an IIOP implementation was available only for the Solaris C++ products. The interoperability of the IIOP implementations provided by the various vendors was the subject of some testing, as documented in Section 3.4.5.

IDL Bindings. The three vendors take distinctly different approaches to support for bindings to different languages.:

HARDPack provides different IDL preprocessors for language bindings in C, C++, Ada95, and Ada83. The stub and skeleton code generated by these preprocessors all bind to a common ORB implementation. LMFS indicates that preprocessors for additional language bindings can be developed in response to customer demand.

ORB*express* provides capability to bind with both C++ and Ada95 applications, but this capability is achieved by providing different ORB implementations for each language. Thus Ada applications bind with an Ada ORB and C++ applications bind with a C++ ORB. OIS indicates that it is considering adding Java bindings but that it will do so, if at all, through implementation of a third (Java) implementation of their ORB.

TAO is a C++-only product, and OCI/Washington University anticipate that this will continue to be the case in the foreseeable future. In response to a direct request about the advice this vendor team would give to a project seeking an ORB but committed to Ada, the OCI/Washington University team responded that they "advocate the use of a mix of ORBs that communicate via IIOP." With regard to plans to support language bindings other than C++ in the future, the OCI/Washington University team responded: "Several customers have requested estimates for providing Java bindings. So far no-one has determined if the benefits of using a single ORB justifies the cost. Mixing various Java ORBS (such as VisiBroker for Java, Sun's Java IDL, and other Java ORBs) and the TAO C++ ORB has not been a problem in practice. However, as the Real-time Java specification effort matures, we will be tracking this standard carefully." They also stated that they would build products for Ada only if a customer agreed to fund the effort.

3.1.3 Self-Assessment with Respect to RT TWG Requirements

The technical self-assessment questionnaire was generated from the list of distributed computing requirements recognized by the RT TWG as of July 1, 1999. Table 3-2 summarizes the responses by vendors when asked for self-assessment with respect to RT TWG requirements. These entries are extracted and paraphrased from vendor written responses to the questionnaire that appears in Appendix A of this document. Final updates to the text of the table reflect vendor reviews of early drafts of this report.

Table 3-2. Responses to Vendor Self-Assessments Against RT TWG Requirements (1 of 3)

Requirement	HARDPack	ORBexpress RT	TAO
(1) Is the ORB designed so that other processes do not block ORB operations indefinitely?	• Operation time-outs have been implemented	• Priority inheritance from user tasks/threads • Data structures that support bounded execution times	• Priority inheritance locks • Standard CORBA messaging timeouts
(2) Does the ORB support transport protocols other than TCP/IP?	• Implemented: – UDP – Shared memory – IP multicast – Unnamed pipes • Planned: – User defined protocols: extensions that allow the using organization to substitute alternative protocols and/or protocol implementations for those supplied with the ORB	• Offers a replaceable transport feature through which users may replace protocols supplied with the ORB: • Implemented: • IP Multicast • UDP • In progress: • Shared memory • Systran Scramnet reflective memory • VME • Planned: • ATM • IEEE 1394 • Fibre Channel • VIA	• TAO provides a pluggable protocol framework • Implemented: • Unix-domain sockets • VME • In work: • TP4 • Fibre Channel • ATM • VIA
(3) Do client requests have an associated priority so that the server can respond to the client request accordingly?	• Client requests under HARDPack have an associated priority by default. Clients can change the priority level to a specific priority for a connection to a server via the *setsched* API. If this API is not called, a default priority is used. In addition, the user can specify the scheduling policy that is associated with the connection using the *setsched* API. • Server can inherit the priority of client. This feature is manually activated using –ORBpriority argument to the call CORBA_ORB_init...	• Priority is inherited from user thread. • The client thread priority is respected and propagated in both local and remote invocations per the OMG Real-Time CORBA 1.0 specification.	TAO does not currently provide propagation of client priority but will implement this capability soon.[3]
(4) Global recognition and communication of priority	Has a priority mechanism that causes a low priority server handling requests from a higher priority client to increase its priority to that of the client	Scheduled for next release.	Scheduled for December 1999
(5) Does the ORB have features that contribute to the desired system level deterministic behavior [beyond requirements noted]?	• Encapsulated Scheduler provides APIs that support building of an RT Scheduler • HARDPack also provides a broadcast capability to achieve timely one-to-many communications	• The RT versions provide for the predictable pre-allocation, allocation, reuse, and priority respecting protection of resources.	____

[3] Note: Per information received 11/19/1999 TAO now provides client-side priority propagation and supports the server-declared model per the RT CORBA specification.

Table 3-2 Responses to Vendor Self-Assessments Against RT TWG Requirements (2 of 3)

Requirement	HARDPack	ORBexpress RT	TAO
(5) continued	• Communication of client request priority to server	• Each component was implemented with predictability and priority management as a prime design goal. • ORBs are designed to scale gracefully and predictably to large numbers of servants (CORBA objects), threads, interfaces, operations, and data volume. • Uses a deterministic marshalling mechanism that both scales well to multi-megabyte invocations and uses memory in a predictable fashion. • Includes a highly predictable, scalable request demultiplexing implementation.	
(6) Does the ORB support bypassing the marshal/de-marshaling of data when the communication is between objects that are coded in the same language and running in the same process?	The user may bypass marshaling by using a CORBA_Any with typecode "TK-void"	Yes	Yes
(7) Does ORB provide the ability to use Quality of Service (QoS) capabilities to provide Guaranteed QoS as defined in IETC RFC 2212?	No	The replaceable transport feature of the RT versions (available for C++ by end of 1999 and for Ada in early 2000) of ORB*express* provides for the definition and use of transport specific QoS. This feature and other controls within the ORB allow the application engineer to select the appropriate quality of service for each request.	End of 1999: ability to use Protocol Properties with RSVP, DiffServe, etc.
(8) Asynchronous Message Architecture	HARDPack provides a subset of Asynchronous Messaging APIs: call back and polling	Partial – yes; Full – future.	TAO supports the callback model for the Asynchronous Messaging Interface of the CORBA Messaging Specification
(9) Does the ORB provide mechanisms to avoid priority inversion in ORB operations?	Server inherits priority of client, manually activated using –ORBpriority argument to CORBA_ORB_init...	RT version of ORB*express* provides special mutex implementations for the protection of ORB and application level resources that avoid both local and distributed priority inversions. These special mutexes are appropriately applied in the ORB*express* RT ORB implementation along with internal algorithms and architectures designed to avoid or bound priority inversions and avoid priority-related deadlock.	• Avoid locking • Use priority inheritance locks when available from OS/RTOS

Table 3-2 Responses to Vendor Self-Assessments Against RT TWG Requirements (3 of 3)

Requirement	HARDPack	ORBexpress RT	TAO
(10) Reliable/unreliable unicast and multicast implemented such that end-to-end latency, latency jitter, and CPU at endpoints scales no worse than linearly with both endpoint load and system-wide traffic	• Default TCP • Optional UDP • Broadcast Service option	Achievable through transport customization	• TCP • Event Service • IP Multicast and UDP broadcast also supported
(11) Memory footprint ... needs to be as small as possible**	Interface Repository can be eliminated to decrease footprint.	• The 2.1.2 version of ORB*express* GT for C++ is 168K bytes of PowerPC object code for the entire ORB. The ORB*express* GT and RT segmentation features allow configurations to remove up to one third of this size. In addition, the ORB*express* IDL compiler is designed to generate memory efficient application code. Future revisions will introduce additional features for memory conservation. • The 2.0.1 version of ORB*express* for Ada is approximately 2M bytes of PowerPC object code for the full ORB. Future versions of the Ada ORB will introduce segmentation features and will significantly reduce the ORB footprint.	TAO supports the minimum CORBA spec; some subsets documented
(12) Does the ORB provide support for persistent bindings?	No, although specific object locations can be specified.	Known endpoints can be supplied.	POA persistent object references supported.
(13) Scalability	"Performance demonstrates scalability" (quoted from vendor response)	Scalable for: • Large sized objects • Large #'s of objects • Large #'s of operations • Large #'s of interfaces • Large #'s of threads • Large #'s of clients • Large #'s of servants	Optimization principle patterns and constant time data structures

3.1.4 Additional RT Extensions

In response to our questionnaire, vendors also volunteered information about proprietary capabilities and services offered in their CORBA products. This information is summarized here. These vendor initiatives may be useful in attempting to predict possible future extensions to OMG standards, since all three of the ORB vendors are active participants in the OMG standards process. However, users must be aware that dependence on proprietary capabilities and services limits the opportunity to port an application to a different CORBA product. Reuse of mission applications across systems is a goal of the DII COE. Application developers should be sensitive to design decisions that create dependencies on specific CORBA products, since the choice of ORB should, if possible, be left to the integrator of applications rather than the developer.

Table 3-3 shows the proprietary capabilities and services offered for each of the ORBs.

Table 3-3. Proprietary Capabilities and Services

Capability/Service	HARDPack	ORB*express*	TAO
RT Events	No	Future	Yes
RT Scheduling Service	No	Yes (available from a third party)	Yes
Debug	Yes	Yes (ORB*express* contains an operation tracing facility)	?
Broadcast	Yes	Future	Yes
Remote Storage Server	Yes	Future	?
Fault Tolerance Services:	Yes	Future	?
• Common Object Mgmt Service	Yes	Future	?
• Recoverable Naming Service	Yes	Future	?
• LAN Switch Manager	Yes	Future	?

Since vendors voluntarily identified these capabilities in response to the initial questionnaire, followup questions were directed to each vendor to try to identify areas of common but proprietary capability. A "?" in the table indicates that we have received no input from the vendor to either confirm or deny support for a comparable but proprietary capability.

[Note: Table 3-3 was updated on 17 November, 1999 to reflect the planned incorporation of existing HARDPack services into the ORB*express* product line.]

3.1.5 Miscellaneous Observations by Evaluators

Some characteristics of particular products were identified as side effects of the direct questions asked, issues raised, and tests performed. Observations that the team felt worthy of note and for which no better home could be found are noted here. Most often these observations will address unexpected limitations discovered during the evaluation process. These notes are organized by product.

3.1.5.1 HARDPack

Problems in use of HARDPack are documented throughout this report.

3.1.5.2 ORBexpress

License keys and license management: Since the ORB*express* evaluations were conducted using demo licenses granted by OIS, access to ORB*express* was regulated through the use of expiring keys for the demo licenses. In addition to access to IDL compilers, the license keys also restricted execution of ORB-based applications in the target environment.

Use of expiring licenses in mission critical systems is prohibited based on requirements established by the RT TWG. OIS indicates that their products currently bypass the licensing logic in the ORB when non-expiring licensing keys are used. In response to a request by the IPT, OIS agreed to produce separate versions of non-evaluation versions of their products that do not contain any licensing logic in the ORB.

3.1.5.3 TAO

The open source model used to develop and maintain TAO results in a product that evolves rapidly. This rapid evolution raises some questions about configuration and quality management. We worked with two versions of TAO in our testing: 1.0.1 (against which results are reported) and 1.0.6, which was downloaded for experimentation late in the test exercises. We found that performance with 1.0.6 was somewhat degraded from our 1.0.1 tests, particularly so for the "Any" tests. This discovery prompted a discussion of the types of testing that are performed against releases of TAO that are available directly from the Washington University web site. While functional regression testing of updates is performed on a regular basis, we were told that extensive performance testing is conducted less frequently. Performance issues are typically resolved only in conjunction with major releases such as those coordinated for more formal release with OCI.

Potential users of TAO should be aware of this "function first, performance later" philosophy and its impact on selection of releases for use in development, test, and fielded systems.

3.2 Connection and Concurrency Models

Our current testing does not address real-time characteristics of ORB behavior. We include this discussion of connection and concurrency models to familiarize the reader with the structure of each ORB, since these aspects strongly influence real-time behavior. The ORB vendors have chosen different approaches to connection and concurrency models within their ORBs.

3.2.1 HARDPack

Changes in the HARDPack connection and concurrency models were in work during the generation of this report. Because the product has been withdrawn as a commercial offering, we did not attempt to track those changes here.

3.2.2 ORBexpress

OIS provides the following description of its connection and concurrency management policies:

> "*Connection Management:*
>
> *"All* ORB*express ORBs establish one connection between a given client and a given server for all the threads between the two, (i.e., a shared connection). There are important scaling issues associated with this decision: the number of connections (and resources associated with each connection) scales with the number of client processes rather than the number of client threads. The cost of each connection is pretty high, so this can be an important economy.*
>
> *"Caveat: In a RT environment and in order to be compliant with the RT CORBA specification,* ORB*express must create additional connections for each CORBA priority range for each server with which a client communicates. These extra connections make the sharing of connections even more critical across different threads. The* ORB*express ORB automatically manages the selection of connections for the client, choosing the connection with the appropriate priority for the request using the active priority of the requesting thread at the time each request is made.*

"By controlling the ranges, the application developer controls the actual connection architecture.

"Concurrency Management:

"By default, the ORB*express ORB assumes a thread-enabled application, i.e., it assumes that it may call application impl[ementation]s from multiple threads, so the application developer must provide reentrant programs.*

"A couple of single-threaded modes are offered as options:

- *" 'Only one thread can be executing in the application code at a given time.' In this model, if an application makes a remote call, the ORB will allow other threads to call into the application while the earlier call is waiting for its remote call to complete. Only one thread is be executing simultaneously inside the application, but this doesn't mean that a particular operation executes to completion before another ORB operation begins. This model works very well with components like Xwindows. One of the strengths of this model is that it's hard to create a deadlock. The downside is that the application code still has to be reentrant, cause you never know when the code will be executed a second time. (When you're using CORBA, you don't know at the time the application is built whether or not the next call will be remote. That decision is made only when the physical allocation to machines and processes is made.)*
- *"The second single-threaded mode might be described as 'One thread at a time, period': once an operation is initiated, no other operation will begin until that operation is completed (impl returns). Under this model it's very easy to create a deadlock situation, since you can't bounce into a remote server that subsequently calls back to this server.*

"OIS is actively participating in OMG dynamic scheduling RFP and expects to support replacement of scheduling algorithms for managing resources and thread contention. These features will be available in the RT product.

"[Dynamic scheduling is in the OMG consolidation process with multiple interested parties working on consolidated initial submission.]"

3.2.3 TAO

Per the vendor's response to our questionnaire:

"*By default, TAO supports a Reactive concurrency model and a non-multiplexed connection model."*

[Explanation added from papers referenced by vendor in response[4]]:

> *"In this connection architecture, each client thread maintains a table of pre-established connections to servers in thread-specific storage. A separate connection is maintained in each thread for every priority level, e.g., P_1, P_2, P_3, etc. As a result, when a two-way operation is [invoked,] it shares no socket*

[4] Schmidt, D. C., S. Mungee, S. Flores, Gaitain, A. Gokhale, "Software Architectures for Reducing Priority Inversion and Non-determinism in Real-time Object Request Brokers," to appear in *Journal of Real-time Systems*.

endpoints with other threads. Therefore, the [client] write, [server] select, [server] read, and [client-to-server] return operations can occur without contending for ORB Core resources with other threads in the process.

*"**Advantages:** The primary advantage of a non-multiplexed connection architecture is that is preserves end-to-end priorities and minimizes priority inversion while sending requests through ORB endsystems. In addition, since connections are not shared, this design incurs low synchronization overhead because no additional locks are required in the ORB Core when sending and receiving two-way requests.*

***"Disadvantages:** The disadvantage with a non-multiplexed connection architecture is that it can use a larger number of socket endpoints than the multiplexed connection model, which may increase the ORB endsystem memory footprint. Moreover, this approach does not scale with the number of priority levels. Therefore, it is most effective when used for statically configured real-time applications...which possess a small, fixed number of connections and priority levels, where each priority level maps to an OS thread priority."*

"TAO now optionally supports a multiplexed connection model, which is like the one provided by ORBexpress and has the same benefits in terms of resource sharing. Please see http://www.cs.wustl.edu/schmidt/ami2.ps.gz for an overview of how TAO's multiplexed connection model works. In addition, the next major release of TAO (TAO 1.1, due out in March 2000) will support the complete "explicit binding" connection model defined by the Real-time CORBA 1.0 specification."

[The explanation inserted below addresses a specific variant of possible "reactive" architectures. It is also an excerpt from Schmidt, Mungee *et al* cited earlier. The reader is referred to the additional papers referenced within the quoted text for more in-depth discussion.]

"The `Reactor`-per-thread-priority architecture is based on the `Reactor` pattern[5], which integrates transport endpoint demultiplexing and the dispatching of the corresponding event handler. This threading architecture associates a group of `Reactor`s with a group of threads running at different priorities. ...[The] component in the `Reactor`-per-thread-priority architecture indle multiple pre-allocated `Reactor`s, each of which is associated with its own real-time thread of control for each priority level e.g., P_1, ... P_4, within the ORB. ...

"Within each thread, the `Reactor` demultiplexes [all] incoming client requests to the appropriate connection handler.... The connection handler reads the request and dispatches it to a servant that executes the upcall at its thread priority.

[5] Schmidt, D. C., "Reactor: An Object Behavioral Pattern for Concurrent Event Demultiplexing and Event Handler Dispatching," in *Pattern Languages of Program Design* (J. O. Coplien and D. C. Schmidt, eds), p. 529-545, Reading, MA: Addison-Wesley, 1995.

"Each `Reactor` in an ORB server thread is also associated with an `Acceptor`.[6] The `Acceptor` is a factory that listens on a particular port number for clients to connect to that thread and creates a connection handler to process the GIOP requests.

***"Advantages:** The advantage of the `Reactor`-per-thread-priority architecture is that is minimizes priority inversion and non-determinism. Moreover, it reduces context switching and synchronization overhead by requiring the state of servants to be locked only if they interact across different thread priorities. In addition, this concurrency architecture supports scheduling and analysis techniques that associate priority with rate, such as Rate Monotonic Scheduling (RMS) and Rate Monotonic Analysis (RMA).*

***"Disadvantages:** The disadvantage with the `Reactor`-per-thread-priority architecture is that it serializes all client requests for each `Reactor` within a single thread of control, which can reduce parallelism. To alleviate this problem, a variant of this architecture can associate a pool of thread with each priority level."*

"TAO now supports a new thread pool model in addition to the "Reactor-per-thread-priority" architecture. ... The new model is called the Leader/Followers Thread Pool model, and is based on the Leader/Followers pattern (which is also documented in the URL [cited] above.).

"In this thread pool model, a server application creates several threads, all of which invoke ORB::run(). The ORB will then select one of the threads to wait for incoming requests. This thread is called the leader thread and will process the first request that arrives to the ORB, but before doing so the ORB will selects another thread in the pool to become the leader. In other words the threads in the pool take turns to process the events. This configuration requires the ACE_TP_Reactor, i.e. applications must use the -ORBReactorType tp.

*" **Advantages**: Compared with conventional thread pool models (e.g., based on the Half-Sync/Half-Async pattern that's documented at http://www.cs.wustl.edu/~schmidt/PLoP-95.ps.gz) the Leader/Followers thread pool pattern improves performance by (1) enhancing CPU cache affinity and eliminates dynamic allocation and data buffer sharing between threads by reading the request into buffer space allocated on the stack of the leader or by using thread-specific storage memory allocations, (2) minimizing locking overhead by not exchanging data between threads, thereby reducing thread synchronization, and (3) Minimizing priority inversion and non-determinism because no extra queueing is introduced in the server.*

*"**Disadvantages**: The main disadvantage of the Leader/Followers thread pool model is that the I/O connections are responsible for buffering requests while threads in the pool are busy, i.e., there is no ORB-level queueing. Although this model is appropriate for most real-time applications, particularly hard real-time applications, some types of applications can benefit from buffering requests in the ORB. Therefore, the next major release of TAO*

[6] Schmidt, D. C., "Acceptor and Connector: Design Pattterns for Initializing Communication Services," in *Pattern Languages of Program Design* (R. Martin, F. Buschmann, and D. Riehle, eds.), Reading, MA: Addison-Wesley, 1997.

(version 1.1, due out in March 2000) will also support the standard thread pool model defined in the Real-time CORBA 1.0 specification, which supports request buffering.

"[TAO] defaults van be modified by providing command-line options or service configuration options to TAO when CORBA::ORB_init() is called by a user. [See] http://www.cs.wust.edu/~schmidt/ACE_wrappers/TAO/docs/configurations.html and http://www.cs.wustl.edu/~schmidt/ACE_wrappers/TAO/docs/Options.html for an overview of the configurations supported by TAO. This information is also available in the TAO Programming Manual that is available from OCI."

3.3 Summary of Product Availability by Platform and Language

There are two aspects to this part of the study. First is availability of the products in both Ada95 and C++ versions for LynxOS on the Power PC, the platform that has been selected as the reference implementation for DII COE RT Extensions. Also of interest to the RT DII COE community are availability on Sun Solaris, a commonly used DII COE platform, and VxWorks, a commonly used RTOS. The other aspect of this part of the study is the ability of the RT CORBA products to facilitate applications independence from the computing platform and operating system. For this aspect we were interested in determining the number of available ports for each RT CORBA product and the effort required to implement a new port.

3.3.1 Current Product Availability

We asked the three vendors what operating systems, hardware, and languages/compilers their products currently support. We updated their responses with vendor review comments plus the ORB*express* information available at http://www.ois.com/ports/ports.htm and the TAO information available at http://www.cs.wustl.edu/~schmidt/TAO-versions-i.html on 18 October 1999. HARDPack information was updated with the HARDPack 1.3 release notes dated 15 October 1999. Available (A) and planned (P) ports with Ada IDL Bindings are listed in Table 3-4. Ports with C++ IDL Bindings are listed in Table 3-5. The most common development platforms for all of the products are Sun Solaris and Intel X86/Windows NT.

Table 3-4. Planned and Available Ada95 Ports

Target Platform	Ada Compiler	HARDPack	ORB*express* RT
DEC Alpha / DEC Unix 4.X	ACT GNAT		A
DEC AXP/ Unix 4.X	Rational Apex		A
HP-PA / HPUX 10.20	ACT GNAT		A
IBM RS/6000 / AIX	(not specified by vendor)	A	
Intel X86 / Windows NT or 95	Aonix ObjectAda		A
PowerPC/ LynxOS	Green Hills		P
PowerPC/ VxWorks	Green Hills		A
PowerPC/ VxWorks	Rational Apex		A
PowerPC/ VxWorks	(not specified)	P	
PPC / LynxOS	Rational Apex	A	
Sun SPARC / Solaris	ACT GNAT		A
Sun SPARC / Solaris	Aonix ObjectAda		A
Sun SPARC / Solaris	Rational Apex	A	A

A = available, P=planned

Table 3-5. Planned and Available C++ Ports

Target CPU-Hardware / Operating System	C++ Compiler	HARDPack	ORB*express* GT	TAO
PowerPC / LynxOs	GNU GCC	A	A	A
Compaq Alpha / DEC-tru64	DEC C++		A	A
Compaq Alpha / Linux	DEC C++			A
Hewlett Packard HP PA / HP-UX	HP aC++		P	A
IBM RS/6000 / AIX	IBM C++	A	P	A
Intel X86 / Linux (Redhat)	GNU GCC		A	A
Intel X86 / LynxOS	GNU GCC		A	A
Intel X86 / QNX	Sybase Watcom C++		P	
Intel X86 / QNX Neutrino	Sybase Watcom C++		P	A
Intel X86 / VxWorks (Wind River Systems)	GNU GCC		A	A
Intel X86 / VxWorks	Green Hills C++		P	A
Intel X86 / Windows NT or 95	Microsoft Visual C++		A	A
Intel X86 / Windows NT or 95	Borland C++			A
Mercury / Mercury OS	Mercury C++		P	
Motorola 68000/ LynxOs	GNU GCC		P	P
Motorola 68000/ VxWorks	GNU GCC		A	P
Motorola 68000/ VxWorks	Green Hills C++		P	
PowerPC / PowerMAX (Concurrent Computer)	Concurrent C++	A	P	
PowerPC / pSOS (Integrated Systems, Inc.)	Diab Data-SDS C++		P	P
PowerPC/ OSE (Enea OSE)	Diab Data-SDS-C++		P	
PowerPC/ OSE (Enea OSE)	Green Hills C++		P	
PowerPC, IBM / AIX	IBM C++		P	A
PowerPC / VxWorks	GNU GCC	P	A	A
PowerPC / VxWorks	Green Hills C++	P	A	A
Silicon Graphics MIPS / IRIX	SGI C++	A	A	A
Sun SPARC/Chorus Classic	GNU GCC			A
Sun SPARC/Solaris	GNU GCC		A	A
Sun SPARC/Solaris	Sun C++	A	A	A
Multiple platforms / multiple OS	Green Hills C++		P	
Multiple platforms / Windows CE	Microsoft Visual C++		P	P

A = available, P=planned

3.3.2 Additional Ports and Upgrades

In the survey we also asked vendors to indicate: (1) the number of weeks after a major operating system upgrade it takes for release of a compatible upgrade of their CORBA products; (2) the time required to port to a new operating system if a customer ordered a version using a language and platform that were already supported; (3) and who would pay the non-recurring costs of a new port. Table 3-6 summarizes the responses.

Table 3-6. Portability Comparison

	Roll to OS upgrade (currently supported OS)	Port to new OS	Responsibility for Non-recurring Costs
HARDPack	On customer demand; no timeline given	Few weeks for C Few weeks + for C++, Ada	Customer
ORB*express* RT	4 - 8 weeks on customer demand	30-60 days	Ports to popular platforms are usually funded in whole or in part by OIS. Ports to less popular platforms are typically funded by customers.
TAO	6 - 8 weeks	< 3 mos. (based on historical max)	Customer (+1 year support if user base is small)

These responses provide little vendor or product discrimination, with the exception of the absence of an LMFS commitment to track operating system upgrades. User responses, as reported in later sections of this document, provided no specific insights into vendor capability to deliver in these time frames.

3.3.3 General Portability Assessment

Based on the portability information gathered for this study, we found that each of the products had advantages and disadvantages.

HARDPack. HARDPack has two points in its favor. First, it is available for Ada and C++. Second, it has ports for LynxOS on the Power PC, and Sun Solaris and is planned for VxWorks. However, it has the fewest total ports. It should also be noted that HARDPack is currently only available for LynxOS 2.5.0, even though LynxOS 3.0 was released in May, 1998. The "Committed HARDPack Development Plan", reference 3, included an upgrade to Lynx 3.0 in the third quarter of 1999. This upgrade was not yet available in the October 15 release of HARDPack 1.3.

ORB*express.* ORB*express* has been ported to a wide variety of operating systems and hardware platforms. ORB*express* is available with Ada and C++ IDL Bindings for Sun Solaris and VxWorks. The ORB*express* for LynxOS 3.0.1 C++ version for Sun/Solaris host was released in September, 1999. A LynxOS for Ada port is not available but will be developed if ordered. OIS expects this to be a straightforward port since the platform, operating system, and language are already supported.

The ACE ORB. TAO has also been ported to a variety of platforms and operating systems, including LynxOS, Solaris, and VxWorks. TAO's biggest drawback is that no Ada support exists or is planned. This clearly fails to meet a RT TWG requirement. However, DOD programs without Ada requirements have been implementing systems using TAO, so it deserves serious consideration.

3.4 IPT Test Description and Results

In this section we describe the independent tests run by the IPT and the results observed. First, the test environment is described and some analysis of the accuracy of measurements in this environment is presented. The results of three test series, Basic Data Integrity (BDI), Basic Interface Definition Language (IDL), and Interoperability tests are reported in sections 3.4.3, 3.4.4, and 3.4.5, respectively.

3.4.1 Test Environment: IPT Integration Facility

Boeing configured a Kent, WA integration facility as shown in Figure 3-1 to perform the RT DII COE benchmark tests. Each of the Sun SPARC development stations (not shown) and Sun SPARC Ultra-1 target stations ran Solaris 2.6. The Motorola MV3600-2 PowerPC (PPCs) used LynxOS 3.0.0, while the Cetia PowerPC was loaded with LynxOS 3.0.1 (ppc-vmpc4a.LynxOS.3.0.1-031). Key characteristics of these computers are noted in Table 3-7.

Unless otherwise noted, all network tests used the TCP/IP protocol over a 10Mbit per second Ethernet. The choice of network options was made for commonality: all of the computers of the testbed support the 10BASE-T (10 Mbit) Ethernet and all of the ORBs operated over TCP sockets.

Table 3-8 lists the software configurations available for the ORB tests.

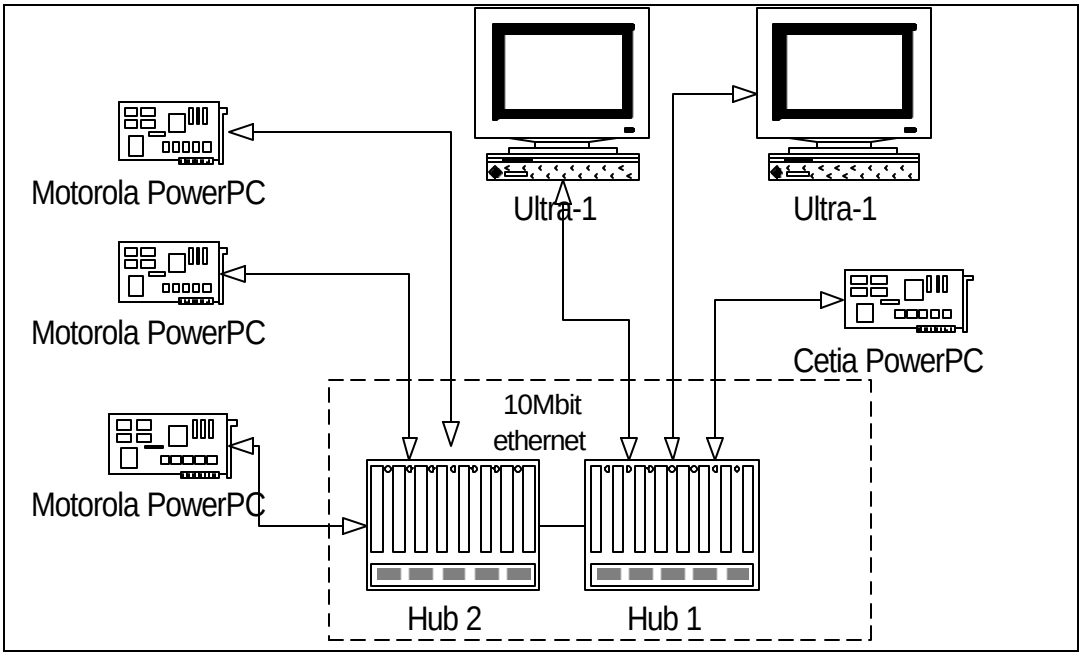


Figure 3-1. Test Bed Configuration

Table 3-7. Selected Characteristics of Benchmark Hosts

Benchmark machine	OS	MPU	Clock speed (MHz)	RAM (MB)	L2 cache (KB)
Sun Ultra1 - #1	SunOS 5.6	UltraSPARC	167	256	
Sun Ultra1 - #2	SunOS 5.6	UltraSPARC	167	128	
Motorola MV3600-2 #s1-3	LynxOS 3.0.0	PowerPC 604	300	256	512
Cetia VMPC5a	LynxOS 3.0.1	PowerPC 750	366	128	1024

Table 3-8. Basic Test Configurations

	HARDPack	ORB*express*	TAO
ORB	HARDPack 1.2.1 (patch release) Also tested 1.1 and 1.2.	• ORBexpress GT 2.1.2 for Solaris/ C++ • ORBexpress GT 2.1.4B-BETA for LynxOS /C++ • ORBexpress RT 2.0.1 for Solaris / Ada95	TAO version 1.0.1 ACE version 5.0.1
Ada	Ada95 Rational APEX version 3.0.0.b plus patches	• Ada95 Rational APEX 3.0.0.b plus same patches as for HARDPack • gnat 3.10	Not available
C/C++	Sun Workshop C++ version 4.2 Lynx G++/gcc version 2/7-96ql	• Sun Workshop C++ 5.0 • Lynx g++/gcc 2.7-97r1	Sun Workshop C++ version 4.2 Lynx g++/gcc version 2.7-97r1
Other	Libraries HARDPack SunOS 5.6 (SunOS 5.5.1 used in tests prior to June, 1999) Lynx 2.5.0		Configuration Files $ACE_ROOT/ace/config.h $ACE_ROOT/include/make/include/platform_macros.gnu Compilation Flags SunOS “-fast –O” LynxOS “-O2”

3.4.2 Dealing with Clocks and Time

3.4.2.1 Local Clock Accuracy

We ran a MITRE benchmark, bintime, to understand the accuracy and variability in clock readings we could expect on our target platforms. The benchmark reads the system clock repeatedly and reports, among other statistics, the smallest difference observed between consecutive clock samples. For the Solaris 2.6/UltraSPARC platform with no network traffic, we calculated over 3,000,000 samples, an average access time of 1 microsecond with an observed maximum of ~1.3 milliseconds. For the LynxOS/PowerPC combination the average access time was 1.8 microseconds, with a maximum observed latency of 22 microseconds.

The operations we planned to measure ranged from a few hundred microseconds to several tens of milliseconds, so the clocks were sufficiently accurate to provide the “ballpark” guidance intended by these benchmarks.

After establishing the granularity at which we were able to access the clock and observing some aberrations in the clock readings, we still decided to include all observations in our calculations. We made this decision for a number of reasons:

- We reasoned that it was important to observe patterns in anomalies, not to mask them.
- A “bad” clock reading occurs because some other activity of the system intervenes between the time that the clock is physically accessed and the time its value is returned to the requester. Any activity that can disrupt clock access in our test bed could also disrupt clock access in an operational system. We felt it was realistic to include the effects of such disruption in our observation of system behavior.
- Since we preserve data on all the individual transactions used to generate average and standard deviation information, others (or we ourselves) may choose to apply other statistical techniques to the original data at a later date.

3.4.2.2 Clock Synchronization in a Network

In order to obtain useful timing information across the network, it was necessary to synchronize the clocks on all the systems involved. As noted below, absolute time was irrelevant, as long as all the machines were synchronized relative to each other. Obviously, there is a degree of jitter/drift involved which has to be compensated for periodically if all systems are using their own independent clocks to time stamp the events.

The basic requirements and constraints for this feature were:

- The machines have to run on an isolated net for the benchmarks.
- Only relative times are of importance - the machine times only need to be synchronized to each other.
- Synchronization between machines should be stable and accurate enough to permit meaningful "flight times" to be acquired for data exchanges.
- A hardware reference clock was unavailable for use.
- Development time for this feature needed to be kept to a minimum.

The network time protocol (NTP) package was used to attempt to keep the system times synchronized. Based on the experience of other sites queried, it was hoped that this would maintain synchronization well enough to obtain useful timing data between machines. Indications were that we could maintain synchronization to within approximately 300 microseconds or better.

Since a hardware reference clock was unavailable, a spare single board computer (SBC) was selected to act as the reference using its internal clock, an "undisciplined" clock in NTP terminology. All the machines performing timing tests were synchronized to this machine using the xntpd daemon, which empirically appeared to work better than having all the machines act as peers to derive a common standard reference. More testing should be performed to determine if better synchronization can be obtained.

Figure 3-2 displays an NTP synchronization log of 211 `ntpupdate` samples taken from one of the SBCs. It shows its hourly adjustments averaged 127 microseconds, with a standard deviation of 739 microseconds but with peaks as high as 6.4 milliseconds. Other SBCs showed similar synchronization characteristics. Using the `ntpdate` utility in a loop to repeatedly synchronize to the reference did not improve these results, so it was deemed acceptable to permit the `xntpd` daemon to sample periodically in the background, assuming it would have minimal impact on timing results.

In summary, NTP on all machines synchronized using the `xntpd` daemon to an SBC which was not participating in any of the timing tests. The "reference" SBC used an undisciplined clock as a reference. The level of adjustment required at each sync interval indicates that "flight time" data between machines may have a rather large component of error introduced by the system clocks on each machine. In other words, a “bad” clock, one that ran much too fast or much too slow, may have contributed to the inaccuracy of our synchronization. Further investigation is necessary if better inter-machine timing information is desired.

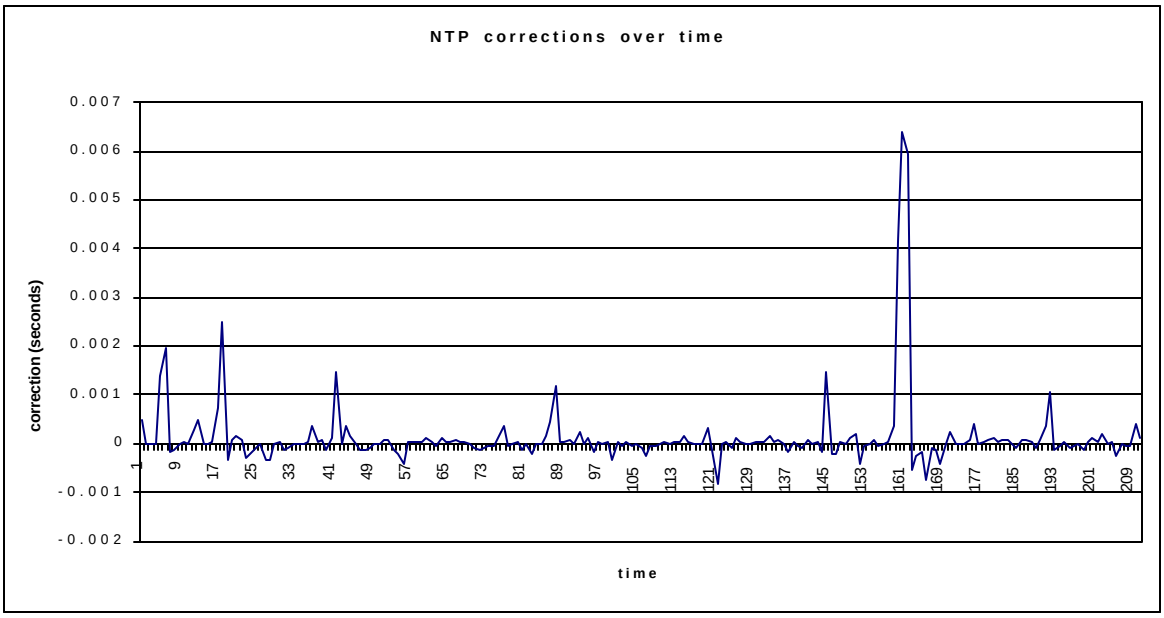


Figure 3-2. NTP Synchronization Log

3.4.3 Basic Data Integrity (BDI) Testing

We designed, built, and executed the Basic Data Integrity test to assure ourselves that each ORB was communicating data with integrity between applications running with different combinations of language, compiler, hardware, and operating system. The test is simple: data structures defined in IDL and containing known values are communicated from client to server and then back to the client. Each participant confirms that the data structure it receives from the ORB contains the expected, predefined set of values. If it does not, the test fails.

The data structure was designed to use a range of data types and alignments that is typical for real-time systems but is only moderately complex. In no way did we attempt to test all possible data combinations.

- The test used the following basic alignment structure:

```
struct basic_alignment {
        boolean b1;
        char c1;
    long l;
        short s1;
        double d;
        char c2;
        short s2;
        unsigned long ul;
        unsigned short us;
        octet o;
        float f;
        char c3;
        boolean b2;};
```

- The test used the following basic enumeration type:

```
enum color {
        red,
        green,
        blue,
        white,
        black};
```

- The test used the following interface structure type:

```
struct interface_struct {
        boolean b;
        basic_alignment ba1;
        char c1;
        basic_alignment ba2;
        color co;
        char c2[2]; };
```

Figure 3-3 shows the control flow for the BDI test. The Client transfers data of the interface_struct type to the Server through the following CORBA transfer methods: (1) Interface structure type, (2) Any, and (3) Sequence interface. The Server receives the data, verifies that all values are received correctly and returns the data back to the Client. The Client verifies that all values received back are identical to those originally sent to the Server. The test is successful if: (1) the data received by the Server has identical values as that transmitted by the Client and (2) the "round trip" data received by the client has identical values as were originally transmitted to the Server. A test failure means either that the data received did not match what was expected or that the test software failed to execute to completion.

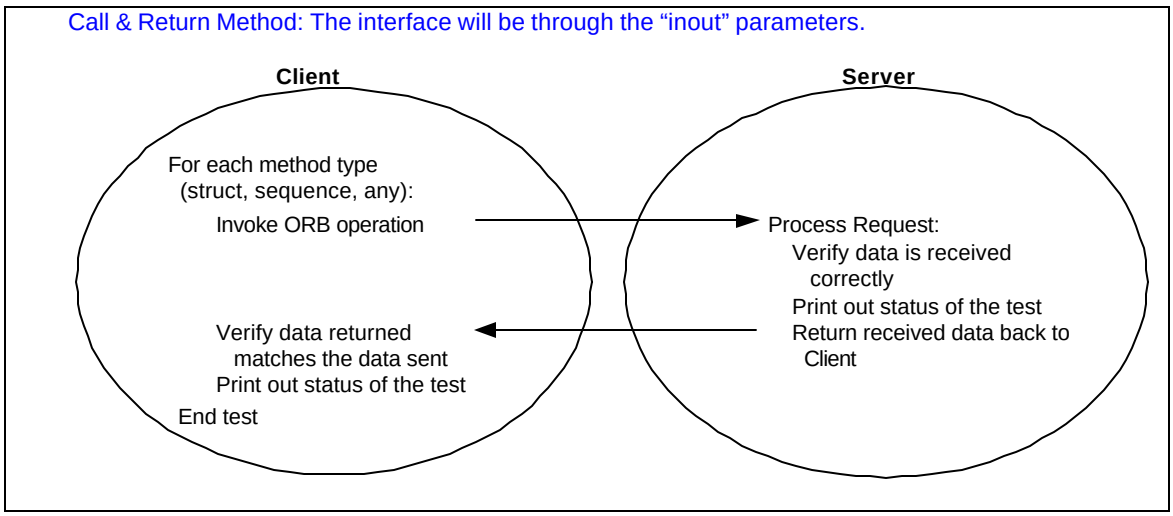


Figure 3-3. Basic Data Integrity Control Flow

Table 3-9 summarizes BDI test results. Combinations of ORB, OS, and language appear across the top of the table as test Client. (In these headings Solaris has been abbreviated to S, LynxOS to L). OS and language combinations only appear on the left as Server. In this sequence of tests, clients using each ORB interacted only with servers using the ORB of the same vendor. (Cross-ORB results are reported in Section 3.4.5, Interoperability Testing.) The Server entries are expanded to include results for each of the transfer types undertaken in the test: IDL structure (St), Any, and Sequence (Sq).

A green entry (G) indicates that the test was passed: both client and server understood the data as expected. A red entry ( ) indicates that the test was failed. A red entry with the notation "cd"

indicates a crash dump. Other failures resulted from receiving data that did not match expectations. Blank entries ( ) indicate tests for which all necessary software is available but which have not yet been attempted. Gray entries ( ) indicate tests which depend on release of software products planned but not yet available. And black entries ( ) indicate a capability that is not currently planned for implementation.

HARDPack failed the BDI test for both cross-language and cross-platform integrity. When the results were reported, the vendor indicated that known problems existed in their capability to handle specific IDL sequences. The vendor says that these problems either have been or will be fixed, but we have run no tests to confirm or deny this statement.

ORB*express* and TAO passed the test in all combinations that were available and tested.

For ORB*express* the tests included cross-language tests on Solaris and cross-platform tests between Solaris and LynxOS with C++ applications code. No product is yet available that supports Ada95 on LynxOS, although this product is planned by OIS.

TAO internal tests were limited, in our test bed, to testing between C++ applications on Solaris and LynxOS. (Ada95 is not supported by TAO nor is support for Ada planned.)

Table 3-9. Results of Basic Data Integrity Testing

			Client											
			HARDPack				ORBexpress				TAO			
			S/ C++	S/ Ada	L/ C++	L/ Ada	S/ C++	S/ Ada	L/ C++	L/ Ada	S/ C++	S/ Ada	L/ C++	L/ Ada
Server	Sun/ Solaris C++	St	G				G	G	G		G		G	
		Any	G	G	G	G	G	G	G		G		G	
		Sq	G	G	G	G	G	G	G		G		G	
	Sun/Solaris Ada95	St		G			G	G						
		Any	G	G	G	G	G	G						
		Sq	G	G	G	G	G	G						
	PPC/LynxOS C++	St			G		G		G		G		G	
		Any	G	G	G	G	G		G		G		G	
		Sq	G	cd	G	cd	G		G		G		G	
	PPC/LynxOS Ada95	St		G		G								
		Any	G	G	G	G								
		Sq	G	G	G	G								

St=structure, Sq=sequence, S=Solaris, L=LynxOS, G=Green, cd=crashed

3.4.4 Basic IDL Benchmarks

The Basic IDL test measures the execution time of individual operations in a quiet execution environment: isolated network, tightly controlled execution activity in which one operation executes at time, unless otherwise noted. The objective of this series of tests was to provide a comparison of basic ORB performance at the level of individual operations.

A key feature of the BasicIDL tests is that, except as explicitly noted, tests were run over a slow (10Mbit per second) Ethernet using TCP/IP as the ORB transport protocol. TCP is notoriously nondeterministic and therefore not suited for achieving predictable real-time behavior. TCP/IP was chosen for these because TCP was **the** transport protocol commonly available among the ORBs under test and, as such, it provided a basis for deriving comparable measurements. The tests were very simple and generally intended to isolate each operation and message transmission from contention for CPU and for network access. However, the use of this protocol in combination with incidental activities of the operating systems undoubtedly contributed some degree of temporal unpredictability that cannot be attributed to the ORBs themselves.

The choice of a relatively slow physical network means that as message sizes grow, the overhead of the network transmission dominates the overhead of the ORB, obscuring some performance differences between ORBs.

Figure 3-4 depicts the general flow of tests to measure the execution time of Call&Return and One-way operations. The client initiated each measured activity by invoking an operation on an object resident in the server. Client and server executed in separate processes. When client and server ran in a single machine, the client ran at a higher priority than the server.

A third process, the background process (or "crumb collector"), ran in the same machine as the client process. The background process "collected" all the "crumbs" of idle CPU time that were available during the test cycle, i.e., it had a priority that was lower than the priority of client and server processes but high enough to ensure that it took precedence over any other incidental activity that might occur. Every effort was made to eliminate any such "incidental" activity during test execution. The number of CPU cycles consumed by the background process represents time not consumed by client and/or server and can be used to estimate the actual CPU consumption of processes running concurrently.

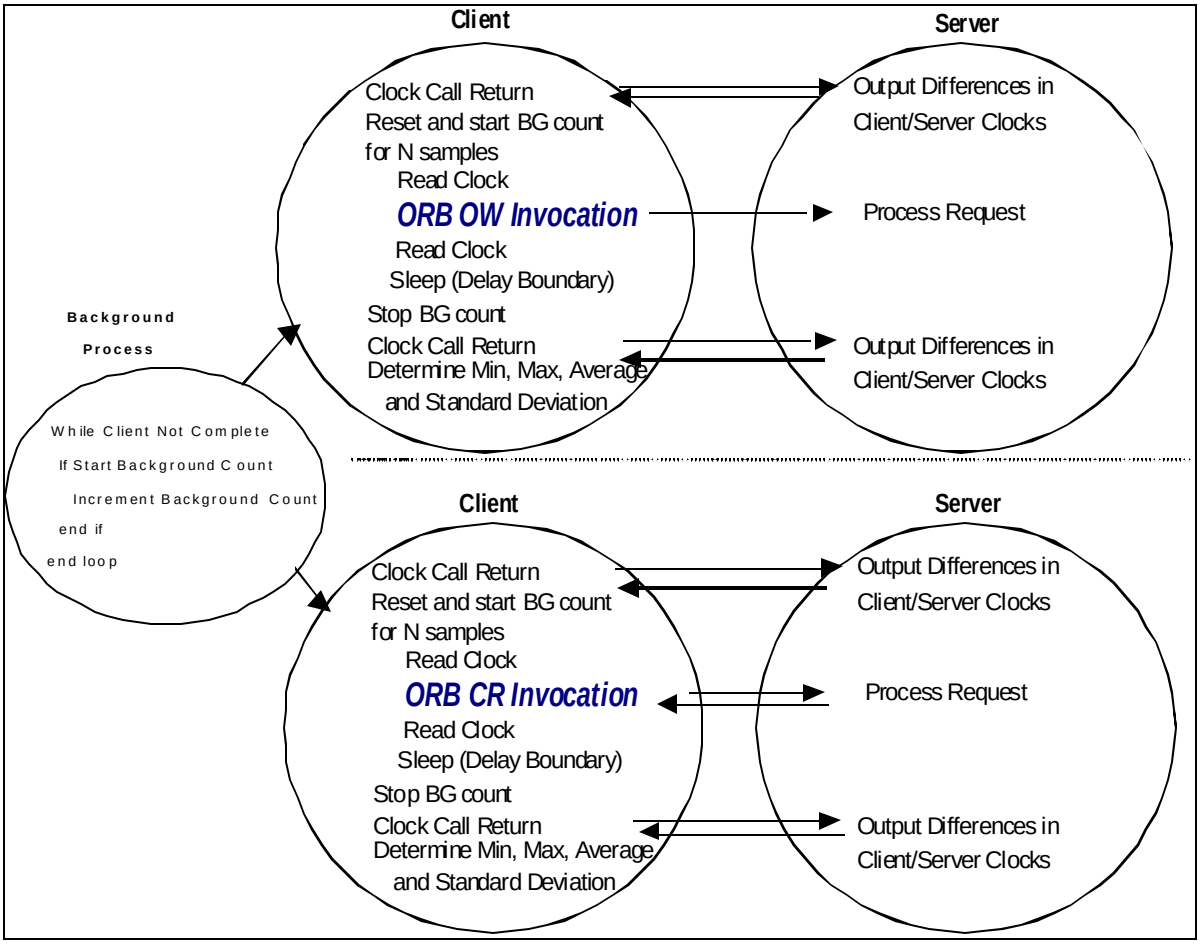


Figure 3-4. Basic IDL Benchmark Test Flow

Under Solaris, client and server ran with real-time priorities. Our intent was to run the crumb collector at the lowest available real-time priority. Under Solaris, however, we experienced some form of catastrophic interaction between the client and server and the crumb collector with all of the ORBs tested, as described in more detail near the end of this section. As a result, the crumb collector ran at the highest timeshare priority under Solaris rather than at a priority of the real-time class.

LynxOS supports a single class of priorities, and all priorities are handled as real-time. No interference between real-time crumb collector and client and server processes was observed in the LynxOS-based tests. Relative priorities were the same in the Solaris tests: client, server, and crumb collector in descending order of priority.

The test invokes a series of data transfers involving a range of data sizes. The data content for the test series is summarized in Table 3-10.

In “official” test runs, each transfer was attempted and measured 100 times. Minimum, maximum, and average values were reported, along with standard deviation. The client-side software measured the time elapsed from the start of the operation call to the return of control to the client. The server-side software recorded the elapsed time from client operation start to receipt by servant. To eliminate I/O operations as a source of interference with test execution, the data for each subtest of 100 cycles was collected in memory by client and server. When a set of 100 samples had been completed for a subtest (a specific instance of transfer type and data message size), the statistics for

the subtest were calculated, and all results written to disk. Detailed records of each individual operation, as recorded at both client and server locations, were also saved for further analysis. An extra call-and-return operation was used to synch client and server execution at the beginning and end of each subtest.

Table 3-10. Test Message Sequence

Attribute	Transfer Method	Data Type	Transfer Units (number of items of the specified data type)
No-Op	Null	N/A	0
Call & Return	No Data	N/A	0
	Short Struct	Short	64, 1200, 2400, 3600, 4800, 6000, 7200, 8400, 9600, 10800, 12000
	Long Struct	Long	32, 600, 1200, 1800, 2400, 3000, 3600, 4200, 4800, 5400, 6000
	Float Struct	Float	32, 600, 1200, 1800, 2400, 3000, 3600, 4200, 4800, 5400, 6000
	Double Struct	Double	16, 300, 600, 900, 1200, 1500, 1800, 2100, 2400, 2700, 3000
	Record Array	Record	4, 75, 150, 225, 300, 375, 450, 525, 600, 657, 750
	Record Array	NA Record	4, 75, 150, 225, 300, 375, 450, 525, 600, 657, 750
	Any	Record	4, 75, 150, 225, 300, 375, 450, 525, 600, 657, 750
One-way	No Data	N/A	0
	Short Struct	Short	64, 1200, 2400, 3600, 4800, 6000, 7200, 8400, 9600, 10800, 12000
	Long Struct	Long	32, 600, 1200, 1800, 2400, 3000, 3600, 4200, 4800, 5400, 6000
	Float Struct	Float	32, 600, 1200, 1800, 2400, 3000, 3600, 4200, 4800, 5400, 6000
	Double Struct	Double	16, 300, 600, 900, 1200, 1500, 1800, 2100, 2400, 2700, 3000
	Record Array	Record	4, 75, 150, 225, 300, 375, 450, 525, 600, 657, 750
	Record Array	NA Record	4, 75, 150, 225, 300, 375, 450, 525, 600, 657, 750
	Any	Record	4, 75, 150, 225, 300, 375, 450, 525, 600, 657, 750

In the initial test cycle the complete BasicIDL test executed in eight different scenarios involving different combinations of Solaris/SPARC and LynxOS/PowerPC (PPC) hosts. Key parameters of the planned scenarios appear in Table 3-11.

Table 3-11. Basic IDL Test Scenario Description

Scenario ID	Host for Client	Host for Server	Frame Time (in msec)
1	SPARC1	SPARC1	0
1a	SPARC1	SPARC1	70
2	SPARC1	SPARC2	0
3	PPC1	PPC1	0
3a	PPC1	PPC1	70
4	SPARC1	SPARC2	70
5	PPC1	PPC2	70
6	SPARC1	PPC1	70
7	PPC1	SPARC1	70

The following rationale explains the choice of some scenario parameters:

Single host tests: Scenarios 1, 1a, 3, and 3a were each run in a single machine. These scenarios served two purposes. (1) They permitted direct measurement of the overhead of operations involving only interprocess communication, i.e., not network transfers, and (2) analysis of the single processor results provide information that was useful in assessing the additional overhead of network based operations. All single host tests utilized TCP/IP as the protocol for communicating between client and server. No tests were run with alternative transports inserted or with TCP/IP bypassed.

"No delay" tests: In scenarios 1, 2, and 3, a new operation was undertaken as soon as control returned to the client from the preceding operation. No artificial delay was introduced between operation attempts.

Each Call & Return operation has the synchronous characteristics of a traditional procedure call: control returns to the caller only when the server completes its computations and returns to the client. The client cannot initiate a "next" Call & Return operation until the current operation completes.

In a One-way operation each client request is queued for asynchronous processing by the ORB and control returns immediately to the caller. Repeatedly initiating additional operations causes request queues to build and lengthens the interval between the request by the client and its receipt and processing by the server. Our "no delay" tests loaded the ORB with a substantial backlog of operations. This kind of pounding is not recommended as an operational mode of behavior, but the tests were useful in helping to identify behaviors exhibited by the ORBs under overload conditions.

Frame-based testing ("with delay" test): In scenarios 1a, 3a, 4, 5, 6, and 7, the test software attempted to initiate operations at a periodic interval of 70 milliseconds, i.e., at the completion of one operation ("sample"), the test software asked that the next operation be scheduled no earlier than 70 ms later than the previous request cycle ("sample") was initiated. We refer to this period or interval as a frame, so these tests were said to have a frame time of 70 milliseconds. The objective was to ensure that each operation executed in isolation with no interference carrying over from earlier operations.

Based on the synchronous Call & Return semantics noted above, the introduction of a measured frame was expected to have no impact on Call & Return (Two-way) operations. For One-way operations, the frame interval nominally provided temporal isolation for each operation. When the execution time of a One-way operation exceeded the basic 70 millisecond frame, this isolation was lost. Although this situation occurred rarely, it is apparent in some of the test results reported later.

Cross-platform testing: Tests mixing Solaris/SPARC and LynxOS/PPC hosts were included as Scenarios 6 and 7 to discover whether heterogeneous operation introduced unexpected performance issues.

The basic tests were extended when it became desirable to explore specific observed behaviors in greater depth. For example, we substituted a Cetia PowerPC executing LynxOS 3.0.1 for a Motorola PPC executing LynxOS 3.0 to explore jitter in the Lynx server.

Test results are available in a variety of formats:

- Comparative results for all (or relevant) ORBs.
  - Average client operation times with error bars showing the magnitude of standard deviations.

- Average client-to-server latency times with analagous display of standard deviation information.
- Trendline equations.

♦ Single ORB results showing the same types of data but in greater detail for a single ORB.
- In addition to average operation times, minimum and maximum times are available.

♦ Detail data: graphs showing individual operation times to help identify the source of summary trends.

Besides raw performance information, we looked for behavioral patterns that might affect the design and implementation of real-time systems:

♦ Predictable operation times

♦ Identifiable patterns in anomalies

♦ Smooth scaling in message / data size

The results presented here are generally drawn from a small number of "official" executions that have been archived for reference. Where irregularities in the data were noted, multiple test instances were run and different permutations of test sequences were attempted to eliminate random environmental perturbation and/ order of execution as a source of the anomalies, when appropriate.

3.4.4.1 Basic Comparative Results

Results from selected individual scenarios are reported in separate volumes. The mapping between test scenarios and volumes appears in Table 3-12.

Table 3-12. Mapping Scenario Summaries to Report Volumes

Scenario	Volume Number
1a – Client and server in single SPARC, 70 ms frame	2
3a – Client and server in a single PPC, 70 ms frame	3
4 – Client and server in different SPARCs, 70 ms frame	4
5 – Client and server in different PPCs, 70 ms frame	5

3.4.4.2 Miscellaneous "Interesting" Observations and Residual Mysteries

LynxOS behavior on large message sizes

When we tested operations involving a server running on LynxOS (scenarios 5 and 6), we encountered much more variability in operation times than we saw in other scenarios. This jitter was evident in transfers of all data types and the variability increased as data / message size increased. We saw the similar variability in operation times across all ORBs and languages in all scenarios in which the server ran under LynxOS.

To make sure that the problem was not attributable to our relatively old release of LynxOS (3.0) or the particular Motorola configuration (PowerPC 604) we were using, we also ran scenarios 5 and 6 under LynxOS 3.0.1, including TCP improvement patches provided by Lynx, on the Cetia PowerPC 750. The jitter persisted in all test cases in which the server ran on a LynxOS/PPC configuration. In cross-platform testing, when the *client* ran on LynxOS and the server on Solaris, the jitter disappeared. We also noted that even with a LynxOS/PowerPC server, the jitter appeared only in client-side measurements: client-to-server latency measurements made at the server in the same test executions did not show similar jitter. The absence of jitter in server-side latencies points

toward some interference in the return of acknowledgements to the client as the source of the variability. We are investigating the configuration of LynxOS TCP in our network environment as a source of these problems and will report on them further as our DII COE RT integration tasks progress.

Ada results including the impact of compiler on benchmark performance

We ported our tests to Ada95 and ran similar scenarios between Ada clients and servers where these capabilities were available. The compilers used for the benchmarks (compatible with ORB and available in our testbed) appear in the entries of Table 3-13.

Although the full results of the Ada tests are available on request, they are not reported extensively here because the performance measurements were made with ORB releases that are already obsolecent and not suitable as a basis for product comparison:

- HARDPack tests were made against the 1.1 release and not updated later because subsequent releases did not feature upgrades to Ada95 capabilities.
- ORB*express* tests were executed against a product architecture that is being replaced while upgrading to the RT CORBA 1.0 standard.

Table 3-13. Availability of Ada95 for Test Scenarios

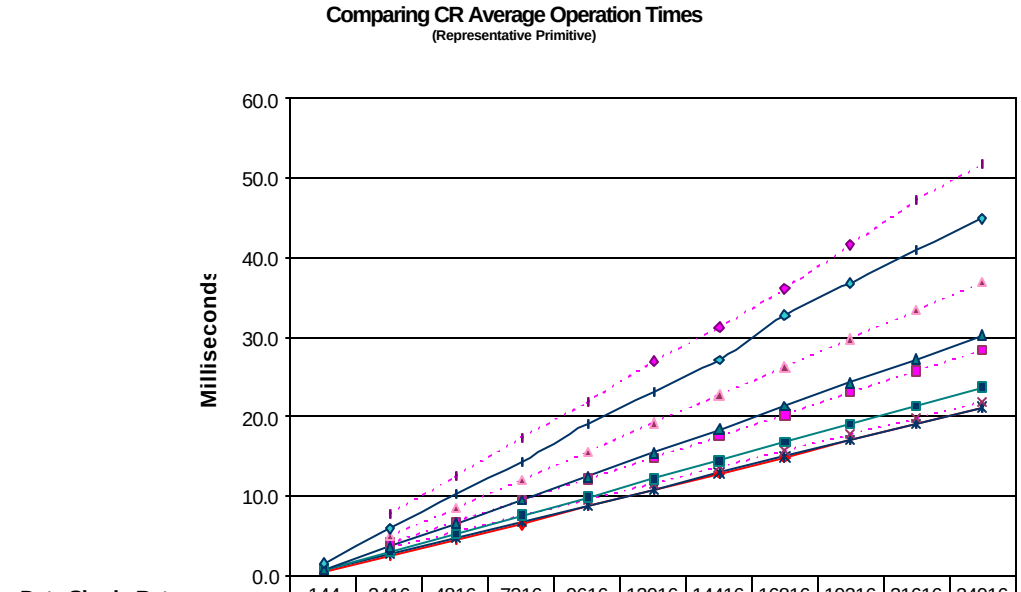
	Platform	
ORB	LynxOS	Solaris
HARDPack 1.1	Apex	Apex
ORB*express* RT for Ada95 2.0.2a	Future	Apex gnat
TAO	Not supported	Not supported

Figure 3-5 compares the average operation times for Scenario 4 (networked SPARCS, 70 ms frame time) over ORB*express* RT for Ada (gnat) with ORB*express* GT for C++. These “old architecture” results for Ada95 are consistently slower than the C++ results, ranging from a few hundred microseconds to milliseconds longer times for operations.

The most significant lesson learned from our Ada95 experiments was the importance of coupling between compiler and ORB. Results for ORBexpress RT on Scenario 4 compiled using the Apex compiler rather than the gnat compiler appear in Figure 3-6. Here performance on records, both aligned and non-aligned, has deteriorated badly, a significant result not anticipated with a simple change of compiler. (This particular interaction of compiler and ORB is being worked as a problem by OIS and Rational.)

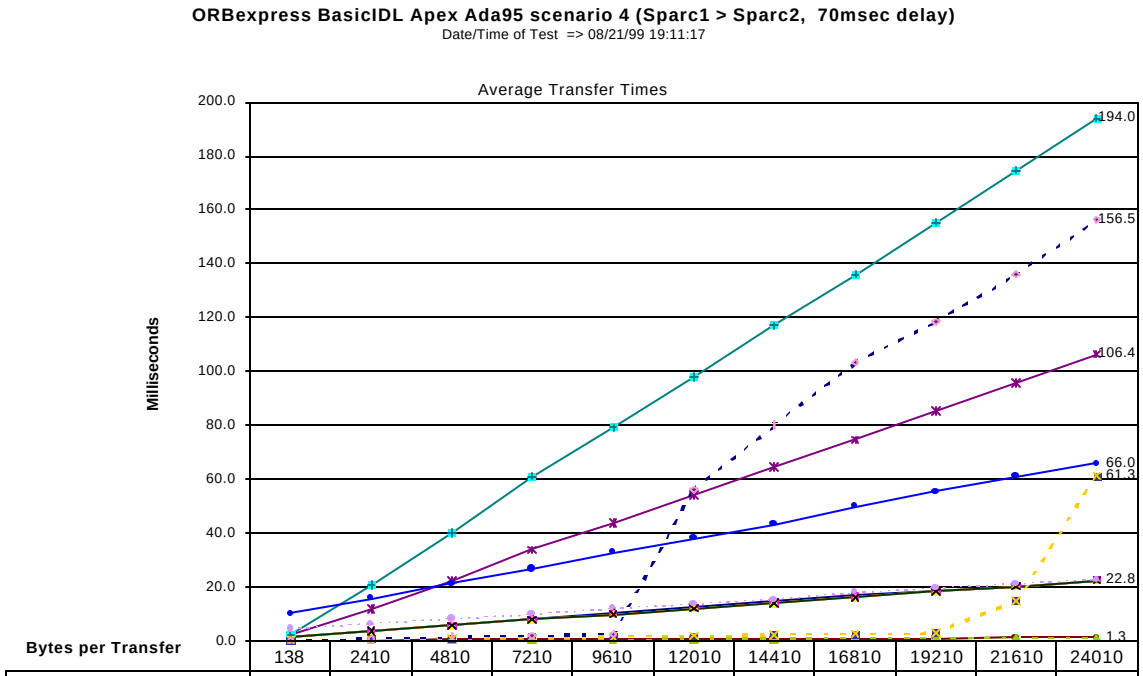
This poor performance on more complex data types was *not* observed in HARDPack test executions tied to the same Apex compiler, indicating that the problem arose from interaction between the compiler and the specific ORB implementation.

Scenario 4: Client, Server on Different Solaris Hosts



Data Size in Bytes	144	2416	4816	7216	9616	12016	14416	16816	19216	21616	24016
Socket	0.5519	2.4713	4.4921	6.4961	8.7675	10.811	12.808	14.862	17.134	19.117	21.166
ORBexpr C++ Float	0.8038	2.7818	4.8346	6.8831	8.8783	10.945	13.019	15.033	17.109	19.106	21.235
ORBex Ada95 Float		3.5263	5.5476	7.5949	9.5875	11.65	13.713	15.76	17.837	19.84	21.912
ORBexpr C++ Record	0.8141	3.0244	5.3368	7.6318	9.9094	12.205	14.513	16.803	19.122	21.392	23.731
ORBex Ada95 Record		4.1249	6.8256	9.4973	12.19	14.853	17.563	20.281	23.107	25.788	28.369
ORBexpr C++ NA Record	0.8442	3.6881	6.6081	9.5838	12.512	15.46	18.43	21.369	24.332	27.282	30.28
ORBex Ada95 NA Record		4.9949	8.508	12.08	15.598	19.283	22.739	26.289	29.872	33.402	36.916
ORBexpr C++ Any	1.6293	5.9299	10.343	14.307	19.122	23.151	27.16	32.814	36.843	40.991	44.926
ORBex Ada95 Any		7.8126	12.522	17.388	21.877	26.998	31.236	36.111	41.618	47.314	51.778

Figure 3-5. ORB*express* Scenario 4: Comparing Ada and C++ on Solaris Platforms



Bytes per Transfer	138	2410	4810	7210	9610	12010	14410	16810	19210	21610	24010
CR Short Struct	1.65124	3.6722	5.81799	8.41142	10.4003	12.5169	14.6089	16.7234	18.3724	20.442	22.5493
CR Long Struct	1.64017	3.66555	5.76945	7.86968	9.91638	12.0362	14.1428	16.2522	18.344	20.4356	22.5442
CR Float Struct	1.65267	3.66907	5.76989	7.87468	9.92047	12.0405	14.1602	16.2333	18.3715	20.4303	22.5386
CR Double Struct	1.65759	3.70934	5.77542	7.87394	9.92659	12.0556	14.1655	16.2316	18.358	20.4446	22.5876
CR Record Struct	2.06671	12.0067	22.5654	33.965	43.8869	54.2454	64.5722	74.8742	85.49	95.8171	106.407
CR NA Record Struct	2.56046	20.8476	40.0244	61.0609	79.2941	98.0565	117.25	136.306	155.352	174.767	193.989
CR Any Struct	10.2352	15.8748	21.5066	26.7378	32.8963	38.0599	43.4335	50.0837	55.4209	61.0699	65.9733
OW Short Struct	0.25669	0.33549	0.43382	0.57113	0.65408	0.7718	0.83962	0.95312	1.07771	1.20147	1.28372
OW Long Struct	0.25753	0.3315	0.44571	0.55435	0.66405	0.77465	0.84404	0.96857	1.0564	1.17202	1.26752
OW Float Struct	0.25794	0.33797	0.4521	0.56645	0.65338	0.76779	0.84004	0.94361	1.08088	1.15108	1.27127
OW Double Struct	0.25663	0.33391	0.44324	0.55694	0.65417	0.77015	0.84242	0.94393	1.0614	1.16788	1.2807
OW Record Struct	0.27184	0.5718	0.93088	1.27912	1.60674	1.95531	2.28803	2.61351	2.98016	14.9464	61.3033
OW NA Record Struct	0.28113	0.83748	1.39086	1.96122	2.51976	56.1246	80.1296	103.458	118.548	136.032	156.519
OW Any Struct	4.5712	6.44087	8.25764	9.87517	12.0756	13.7908	15.4565	18.0197	19.6333	21.2148	22.7976

Figure 3-6. ORB*express* Ada95 Results: Scenario 4 Using Apex Compiler

The Solaris "crumb collector" problem

As noted earlier, we were unable to execute the BasicIDL test precisely as planned because of apparent interference between a crumb collector executed at a lower, but still real-time priority, and real-time client and server on Solaris platforms. Interference from the real-time crumb collector affected tests intermittently and to varying degrees with all three ORBs tested. In the end, we ran and reported "official" results running the crumb collector at the highest time-share priority rather than the lowest real-time priority. Because the interference problem was somewhat intermittent, we were able to capture comparable data for test executions with crumb collector at real-time and time-share priorities. This data showed that the effect of the real-time priority on test data collection was minimal, so "official" tests were subsequently run with the crumb collector running at the highest time-share priority. The "crumb collector problem" remains, however.

The interference between the crumb collector, a computationally intensive, low priority real-time activity, and the client and server manifests itself as either “soft” or “hard” hang-ups of the system at various times. In the soft crashes, the target system continued to respond to “pings” from other machines on the network, but no apparent progress was made on the benchmark tasks. When the failure was hard, the “ping” response disappeared as well.

We originally suspected a simple priority inversion problem in the Solaris kernel that would allow a process with a low real-time priority to withhold the CPU from a higher priority real-time process waiting for completion of a system service. Persistent hangups of HARDPack and ORB*express* tests pointed to a well-defined problem.

In an attempt to isolate the problem to the operating system and eliminate the ORB as culprit, we designed a simple test in which socket-based applications competed with a real-time crumb collector for execution time. Overlapping execution of the processes under test was verified by recording start and stop times of all processes and showing through post analysis that the crumb collector, modified to run for a bounded time, ran to completion within the envelope of execution for the client and server applications. Analysis of cycle times for client and server showed that the execution of the crumb collector had no untoward effect on the execution times of the higher priority client and server, so the problem could not be attributed to a priority inversion that could be triggered simply by inserting lower priority real-time computations.

Because our test software used a more complex shared memory interface for coordination between client and server, we postulated that introduction of the shared memory interactions could have triggered the problem. Adding a shared memory control interface to our simple test, however, failed to produce the desired interference either.

Over time we also found that the problem was more intermittent than originally thought. Unintentional tweaks to our test environment introduced during a move of lab equipment reduced the frequency of the hangups but did not eliminate them. TAO appears to hang up somewhat less frequently than either ORB*express* or HARDPack, but it does hang up on occasion. The intermittent nature of the problem causes us as much concern as the basic problem itself: if we cannot identify the circumstances that trigger the problem, we cannot offer advice on how to avoid it.

We continue to pursue this problem as an issue in the use of Solaris. We plan to run our tests with Solaris 8, which is reported to have improved real-time capabilities, to see if the problem persists with the newer OS release. Since the absence of a failure in a limited number of test executions cannot prove that the problem has been fixed, however, we will continue to investigate the source of the problem with Sun in hopes of assuring ourselves that it has been or can be eliminated in the future.

3.4.4.3 Summary of Basic IDL Performance Results

A summary of results is simple: in general, ORB*express* products outperformed alternatives in terms of both raw speed and predictability in most cases tested:

- **Within a single machine,** ORB*express* **generally outperformed alternative products** in the configurations we tested. We did not test performance optimizations for operations executed within a single process nor did we test any optimizations to bypass TCP/IP using either standards-based or proprietary capabilities. We did observe a performance advantage for ORB*express* in both LynxOS/PPC and Solaris machines for the situations tested.
- **For transfers of primitive data types over the relatively slow network of our test environment, the performance of TAO and** ORB*express* **on primitive data types was**

roughly equivalent and approached the performance of our direct, unmarshalled transfers over TCP sockets for some data sizes. HARDPack performance lagged, but some of the performance degradation may be attributable to a software error reported by the vendor that involved nearly doubling the number of bytes transmitted over the network.

- **For transfers of more complex data structures (including aligned and non-aligned records) over a network,** ORB*express* **outperformed TAO.**[7]
- ORB*express* **performance on Any transfers bettered other products by a wide margin.** The measured operation times for ORB*express* Anys were generally half the time measured for other products or better.
- **The products, in general, produced fairly consistent operation-to-operation performance.** Except for scenarios involving a LynxOS server over a network, there was not a significant amount of jitter in results observed. (This was expected in our controlled, otherwise quiet execution environment.)
- **When timing anomalies existed, the combination of ORB and operating system running over TCP/IP occasionally produced extreme deviations from the expected behavior.** Each ORB produced some test runs in which single-sample anomalies dominated the computed standard deviation, and at times these anomalies were extreme. For example, in LynxOS tests, TAO exhibited anomalies in the largest data sets for operations on non-aligned records. In these scenarios, the "bad" operation times were many times the magnitude of the nominal operation time. (These particular anomalies disappeared in the cross-platform scenarios involving LynxOS and Solaris.) ORB*express* exhibited anomalies with at least one peak of equal magnitude in the Float series of OW operations of Scenario 7 (LynxOS client, Solaris server). In this case, however, the anomalies did not recur consistently in repeated executions of the test scenario.

3.4.5 Interoperability Testing

We expanded our Basic Data Integrity test to cover communication between ORBs. Table 3-14 summarizes the status of these tests. The light blue entries ( ) represent tests that have been run and reported earlier in the section on Basic Data Integrity Testing. The entries with diagonal cross-hatch ( ) denote tests that require software not yet released by a vendor. Specifically the ORB*express* Ada95 ORB for LynxOS is not yet available. White entries indicate tests that we did not have time to run as part of our independent testing activities.

We did not test the interoperability of HARDPack with the other ORBs because our attempts to execute using their IIOP implementation were unsuccessful.

The results of these tests, although incomplete, are encouraging. The IDL used in these tests has already been described in the Section relating BDI results. Reciprocal communications between TAO and ORB*express* tested satisfactorily except when executed using the "Any" method. The immediately positive results are indicated in the table by the light (green) solid entries (G).

Washington University and OIS helped us find an area of the test IDL in which interpretation and implementation of the standard differed. While the exact nature of the specification

[7] HARDPack performance varied relative to the other two ORBS, but its use of proprietary protocols and questions about the integrity of the data transfers tested make it impossible to draw supportable conclusions based on current observations.

interpretation problem was not agreed upon by the vendors in the time frame of our testing, they helped us find an alternative, logically equivalent IDL representation that both handled acceptably. With this change, the tests based on "Any" also ran successfully. Successful execution of a modified test is flagged in the table as a darker (green) solid entry ( ).

There was not enough time to integrate the final IDL changes back into the Orbix and ORBexpress Ada programs and get reliable results. This work will continue and will be reported as part of DII COE RT integration work.

The set of test combinations continues to expand. We also tested ORB*express* C++ and TAO on both Solaris and LynxOS platforms against ORB*express* GT for C++ 2.1.1 executing on a Compaq Tru64/Alpha platform. The test of the Any interface was omitted at the request of the program initiating the testing. All tests attempted executed successfully. These tests are notable for demonstrating that communication between big- and little-endian architectures is handled properly in the interface.

Our conclusions from this testing were generally positive. The ORBs communicated successfully with a relatively small amount of effort on the part of the development team. When problems developed, we got good support from vendors to resolve them. In general, CORBA IDL appears to make the steps of definition and checkout of interfaces easier, but our tests reinforced the lesson that the checkout step is still essential.

Table 3-14. Interoperability Test Results

Server / Client		ORB*express* C++		ORB*express* Ada95		**TAO** C++		**Orbix** C++
		Solaris	LynxOS	Solaris	LynxOS	Solaris	LynxOS	Solaris
ORB*express* **C++** Solaris	ST	✈	✈	✈	✈	G	G	
	Any	✈	✈	✈	✈			
	SQ	✈	✈	✈	✈	G	G	
ORB*express* **C++** LynxOS	ST	✈	✈	✈	✈	G	G	
	Any	✈	✈	✈	✈			
	SQ	✈	✈	✈	✈	G	G	
ORB*express* Ada95 Solaris	ST	✈	✈	✈	✈			
	Any	✈	✈	✈	✈			
	SQ	✈	✈	✈	✈			
ORB*express* Ada95 LynxOS	ST	✈	✈	✈	✈			
	Any	✈	✈	✈	✈			
	SQ	✈	✈	✈	✈			
TAO C++ Solaris	ST	G	G			✈	✈	G
	Any					✈	✈	
	SQ	G	G			✈	✈	G
TAO C++ LynxOS	ST	G	G			✈	✈	
	Any					✈	✈	
	SQ	G	G			✈	✈	
Orbix 3.0 C++ Solaris	ST					G		✈
	Any							✈
	SQ					G		✈

(ST = Structure; SQ = Sequence)

The way in which some of the tests failed caused some concern. We were pleased that none of the three ORBs tested here ever indicated they had communicated correctly when in fact they had not. We were not pleased that the presence of problems in the incoming data formats sometimes caused the receiving program to crash. A more graceful degradation than total failure when a "badly" formatted message arrives is highly desirable.

3.4.6 Other Benchmarks

There are many ORB benchmarks and ORB benchmark results available from a variety of sources. Many of these benchmarks document average performance only without providing the detailed insight into worst case performance that real-time systems need. RT benchmark data is or will be available from other sources, too, including:

- Dr. Douglas Schmidt's research group at Washington University has designed and executed some real-time benchmarks. Their results are reported in various papers, references to which appear on Dr. Schmidt's web page (http://www.cs.wustl.edu/~schmidt/TAO.html). These benchmark programs can be downloaded from the web and reused in different environments.
- Benchmarks that are both comparable and compatible with our own have been run by the Upgraded Early Warning Radar (UEWR) program against HARDPack (DSI) and ORB*express*. The results for the DSI version of HARDPack are documented in *Product Assessment Report of Lockheed Martin's DSI 6.0*.[8] Similar reports are being prepared for ORB*express* and possibly TAO.
- The Naval Surface Warfare Center at Dahlgren has run a number of real-time benchmarks and expects to publish results for HARDPack and TAO in the near future. The benchmarks executed by the NSWC this year augment the information reported here with some tests that demonstrate scaling properties. NSWC plans to expand their benchmark suite in CY2000, and the IPT plans to track these developments.

Our principal contribution is to provide a comparison of like-capability in a controlled and stable execution environment. Where feasible, the RT IPT will provide additional benchmark results, based wherever possible on the programs developed by these and other groups.

3.4.7 Limitations of IPT Testing

The benchmarks conducted by the IPT explore only limited aspects of ORB performance and behavior that should be considered during (1) selection of CORBA as a viable technology for distributed computing infrastructure in a particular system and (2) selection of a particular ORB once CORBA is identified as a desired technology.

- Our tests include no comparison of performance beyond basic IDL interfaces. In particular, we have not explored the performance of features like event services or asynchronous messaging, capabilities that figure prominently in some existing RT system architectures.

[8] *Product Assessment Report of Lockheed Martin's DSI 6.0*. for Task Order 0014 (Common Operating Environment-Risk Reduction ND Analysis (COE-R^2/ND)) for the Command and Control Product Line (CCPL) Program, CDRL Sequence NO. N007, Task Order Description (TOD) 0014, Paragraph 5d "DELIVERABLES", DI-MISC-80711/T, ESC/NDB, Hanscom AFB, MA.

- **Our benchmarks do not address real-time behaviors at this time**. All measurements were taken in quiet, isolated systems in which no attempt was made to gauge the interference with higher priority activities by lower priority activities. The rapid changes that are being made to "real-time" aspects of the current set of products influenced the decision not to compare priority handling at this time. All of the products under consideration are being upgraded to provide some end-to-end understanding of priorities in accordance with the CORBA RT 1.0 specification. Until changes to support this specification are implemented, any results reported on the handling of real-time priorities will be immediately outdated.
- Perhaps most critically, a set of standard benchmarks run in a "vanilla" testbed cannot provide information that can be precisely projected to a specific system environment. A minimal list of factors that influence raw performance include specifics of board and computer configuration; operating system release and configuration; network, network interface card, and network drivers; and choice of compiler. In addition, few programs will find that the data and transfer types represented here have any close correlation with the actual message set that is found in a real system.

In light of these shortcomings, the results presented here should be taken for what they are: a very basic set of tests comparing a minimal set of ORB capabilities. The results reported should help in narrowing a field of choices for a particular program, and they may also influence the priority with which ORB products are considered for use. If performance is a serious concern, benchmark testing should be tailored to the specific operational environment under consideration before binding decisions are made.

3.5 Conclusions and Recommendations

None of the three products we evaluated completely fills the gamut of real-time systems needs. HARDPack meets many of requirements of the RT DII COE community, but for only a narrow range of platforms and often based on proprietary approaches. It performed poorly in our testing, but the vendor has indicated that significant problems affecting both timing performance and integrity have been found and fixed. Unresolved issues such as unavailability of IIOP make it difficult to assess the capability of the product and its utility for the general RT community.

Technically, the ORB*express* and TAO products impressed favorably. The products performed reasonably well, and the vendors demonstrated in-depth knowledge of real-time systems design issues, commitment to standards-based infrastructure, and aggressive support for the performance, capability, and integrity of their products. TAO offers a feature-rich environment in which many services are emerging. It falters, however, in failing to provide Ada95 support, a capability that is critical to many in the RT DII COE community. The emphasis in the ORB*express* product line appears to be on a highly capable core ORB available across a wide range of execution platforms. However, services beyond the ORB core have limited. The recent joint announcement by OIS and Lockheed Martin indicates that OIS is taking steps to change this position in the future.

The ongoing transition of products to the RT CORBA standard means that our analysis is weak in assessment of RT features. The ORB vendors have plans to bring their products into conformance with the standard. After conformance is first achieved, it may be some time before maturity is also achieved. Until mature, conformant products exist, a complete assessment is out of reach.

4. Support Environment Assessments

4.1 Usability and Support Summary from User Questionnaires

User survey data was collected from the organizations listed in Table 4-1 and compiled in late September and early October of 1999. The reader should remember that user survey responses are subjective and are closely coupled to the particular application that the CORBA product user was involved with. Responses are also based on user experience with the products and product support that were available to them at the time and may differ from what is available to the reader of this report. Respondents had varying amounts of experience with the products and different levels of CORBA programming expertise. No attempts were made to differentiate between different versions of the same vendor's product (e.g., ORB*express* RT for Ada and ORB*express* GT for C++). Users of TAO were working with rapidly changing versions of the open source product, not the recently released commercialized version. Finally, due to the small number of responses received, these results are not statistically significant.

Table 4-1. Organizations Responding to User Survey

Responder	Product	Characteristic Execution Environment	Application
Boeing – AWACS program	DSI/HARDPack	S, W	Airborne warning and control
Boeing – Bold Stroke	TAO	H, S, E, W	Fighter aircraft embedded avionics
Boeing – Open System Architecture	ORB*express*	W	ORB evaluation for command & control software architecture
DCS Corporation	ORB*express*	H, S, E, W	Ground combat vehicle electronics
KRONES AG	TAO	S, E	Inspection machine for bottles
Lawrence Livermore National Lab	ORB*express*	S, E, W	National Ignition Facility control system
MITRE – AWACS Step 1 Program	DSI/HARDPack	S, W	Airborne warning and control
NASA Ames Research Center	TAO	S, W	Astronomical data systems
Naval Surface Warfare Center – Dahlgren, VA	ORB*express*, HARDPack, TAO	S, W	CORBA evaluation
Paranor AG	ORB*express*	S, E	Monitoring, control, & alarm system for a twin-hull luxury yacht
Tri-Pacific Software, Inc	ORB*express*, HARDPack	H, S, W	RT CORBA Scheduling Service
U.S. Navy SPAWAR	ORB*express*	S, W	Satellite network management system
Boeing – RT DII COE evaluation team	ORB*express*, HARDPack, TAO	H, E, W	RT CORBA evaluation

H = hard real-time S=soft real-time E=embedded W = workstation

Host and target platforms were predominantly some version of Sun SPARC running some version of Solaris. Table 4-2 lists the environments used by the respondents without distinguishing between different models of a hardware platform or different software versions.

Table 4-2. Environments Used by Respondents

Platform	Operating System	Compiler	No. Users
Sparc	Solaris	Sun Native C++	5
Sparc	Solaris	GNAT Ada95	3
X86	Windows NT	Microsoft Visual C++	3
PowerPC	VxWorks	Green Hills C++	2
Sparc	Solaris	Rational Apex Ada95	2
X86	Linux	GNU GCC	2
Power PC	VXWorks	Rational Apex Works	1
PowerPC	VxWorks	GNAT Ada95	1
Sparc	Solaris	GNU C++	1
X86	VXWorks	Green Hills C++	1
X86	Windows NT	Aonix Object Ada	1

RT DII COE integration environments not included

We asked questions about training and other kinds of engineering services available for separate purchase, but most of the respondents had not used them. Those that had used them were generally satisfied with the experience. We also asked whether or not any of the users had replaced TCP/IP with another transport protocol but no one gave a positive answer.

4.1.1 ORB Attributes

We asked users to rate the features of the RT ORB that they are using in their real-time applications. Responses were scored so that features identified by users as most critical to program success were given 5 points. Those of lesser importance were given scores ranging from 4 to 1. The results, totaled and sorted in descending order (most critical first), are shown in Table 4-3. Other desirable features noted by users were Portable Object Adapter (POA) and Ada83 IDL bindings.

Table 4-3. ORB Feature Criticality to Users

ORB Feature	Score	ORB Feature	Score
IDL Compiler	78	Interface Respository	17
Static Invocation Interface	70	Dynamic Invocation Interface	16
IIOP	62	Messaging Service	16
C++ IDL Bindings	60	Implementation Repository	14
Ada95 IDL Bindings	50	RT Trader Service	13
Naming Service	48	Java IDL Bindings	11
Asynchronous I/O	29	RT Transaction Service	9
RT Event Notification Service	26	Time Service	5
ANSI C IDL Bindings	19		

Attributes identified as influencing ORB selection were reliability, and to a lesser extent, fault tolerance. Write-in influences included broadcast/multi-cast, vendor responsiveness and support, the existence of other users, open source model, and the provision to allow custom transport protocols. ORB*express* users were generally happy with the products, but several mentioned the need to build their own Name Service and Event Services as a weakness. TAO users mentioned real-time optimizations, portability, and configurability as strengths with limited support as a weakness. Seven organizations reported that they have used IIOP to communicate between heterogeneous platforms.

4.1.2 ORB Performance Considerations

We asked users to specify and prioritize the performance drivers in their ORB-based architectures where "1" was most critical to their system. Results are shown in Table 4-4, where "score" was calculated by averaging the mean, mode, and median of the user specified priorities. This score is meaningless except as a rough indicator of the relative importance of these performance drivers to the users that responded to this question.

Table 4-4. Performance Drivers

Performance Driver	Score	Ranges
Average Operations Latency	1.5	Not specified
ORB Operation Rates	1.7	1 to >1000 ORB operations /second
Predictable Operation Latency	1.8	2 to 100 ms
Complexity of Interface Data	2.1	Simple to very complex
Total Data Volume	2.9	2400 bytes to 5MB/second
Typical Message Size	3.9	1 byte to 200Kbytes

Other performance considerations mentioned by users included: Ada to C++ ORB performance, performance of "Any" and data marshalling, inter-ORB clients and servers, extensive use of callbacks, and priority management.

4.1.3 Product Installation and Use

Users were asked for their opinions on the quality and usability of the RT CORBA product. Responses were scored as follows: strongly agree = 2, agree = 1, disagree = -1, and strongly disagree = -2. One responder wrote in "**VERY STRONGLY AGREE"** for one response. This response was scored as "3". Table 4-5 shows the average responses by product, numbers in () indicate the number of responses received. Opinions about TAO reflect use of the open source product, not the commercialized version recently released by Object Computing, Inc.

Table 4-5. User Opinions — Product Quality and Usability

	HARDPack	ORB*express*	TAO
This product is easy to install and integrates easily into our development environment.	0.60 (5)	0.88 (8)	0.00 (6)
This product integrates easily into our configuration management environment.	0.67 (3)	1.14 (7)	0.33 (6)
I am satisfied with the general capability of this product.	-0.60 (5)	1.63 (8)	1.50 (6)
I am satisfied with the real-time features and behavior of this product.	-0.25 (4)	1.43 (7)	1.40 (5)
My organization has found it easy to design and implement software using this ORB.	0.25 (4)	1.38 (8)	0.83 (6)
Compared with other ORBs, my organization has found this product to be easier to use than other products. (leave blank if not applicable)	-1.5 (2)	1.71 (7)	-1.00 (3)

4.1.4 Product Support

We asked users how many problems they had reported to the vendor over the last 3 months , last year, and life of product use. Nine users responded to this question. Numbers of problems in the

last year that prevent or degrade mission accomplishment with no workaround possible (priorities 1 and 2) written against ORB*express* and TAO ranged from 0 to 4. Only one HARDPack user responded to this question. That user reported 40 priority-2 problems. Problems of all other priorities reported for ORB*express* and TAO ranged from 0 to 5. The HARDPack user reported 39 problems.

Users were also asked for their opinions on the quality and usability of the product documentation. Responses were scored as follows: strongly agree = 2, agree = 1, disagree = -1, and strongly disagree = -2. One respondent inserted "very much" in a strongly agree box, this response was given a score of 3. Table 4-6 shows the average responses by product, numbers in () indicate the number of responses received.

All TAO users responded based on use of the product prior to the July, 1999, formal involvement of Object Computing, Inc., in the commercial support of TAO. On 11 October, 1999, Object Computing, Inc. stated that at this time support is being provided on an hourly basis of $125, charged to a credit card. Yearly maintenance contracts are anticipated to be available in February or March of 2000. These cost figures were not subject to user review and comment in this survey, however.

Table 4-6. User Opinions — Vendor Product Support

	HARDPack	**ORB*express***	**TAO**
I am satisfied with the technical content of a typical vendor response to a problem report	0.50 (2)	1.86 (7)	1.33 (6)
I am satisfied with vendor responsiveness to critical problems (priorities 1 & 2).	-0.67 (3)	1.57 (7)	1.40 (5)
I am satisfied with vendor responsiveness to routine problems (priorities 3-5).	-0.67 (3)	1.29 (7)	0.83 (6)
The cost of product support is reasonable.	1.00 (2)	1.60 (5)	2.00 (6)

4.1.5 Documentation

Users were asked for their opinions on the quality and usability of the product documentation. Responses were scored as follows: strongly agree = 2, agree = 1, disagree = -1, and strongly disagree = -2. Table 4-7 shows the average responses by product, numbers in () indicate the number of responses received. Object Computing, Inc. released a set of user documentation for TAO in late September but respondents to this survey probably did not have it.

Table 4-7. User Opinions — Product Documentation

	HARDPack	ORB*express*	TAO
The documentation is technically accurate.	0.4 (5)	1.43 (8)	0.4 (5)
The documentation includes a complete source of reference information.	-0.8 (5)	0.71 (8)	-0.4 (5)
The documentation is clear and unambiguous.	-0.2 (5)	1.00 (8)	0.4 (5)
It is easy to find relevant topics.	-0.2 (5)	1.29 (8)	-0.8 (5)
The documentation includes excellent instructional material.	-1.2 (4)	0.71 (8)	-0.8 (5)
The cost of documentation is reasonable.	1 (2)	1.60 (6)	1.8 (5)

4.2 Local (Team) Usability Assessment

4.2.1 Effort to Port

HARDPack. Since we initiated CORBA benchmark testing using HARDPack, all of the benchmark tests were initially developed for it. In developing the benchmark test software we encountered several HARDPack problems of two basic categories: documentation and software bugs. In the documentation category: (1) the paper and on-line documentation was out of date and incomplete; (2) there was no clear description of the steps required to initialize the server and client; and (3) the descriptions of the APIs and their parameters in the C++ and Ada95 language reference manuals are incomplete and inaccurate. We had to use the examples provided in the documentation to determine how to configure and use the APIs. In the software bug category, we found an undocumented "feature" on the SPARCs where the ORB would change the process priority from real-time to time-share if the thread priority was left to the default during initialization. This problem only occured in the C++ version. The default in Ada was such that the process priority was not modified, and it took a HARDPack engineer looking at the HARDPack source code to resolve the problem.

The port to the PowerPC/LynxOS was also difficult. The C++ BasicIDL benchmark test software must be broken up into small executables in order for it to initialize correctly. In compiling the C++ benchmark test software for the PowerPC/LynxOS we also discovered that in one of the C++ HARDPack header files the word "Status" was defined as "void". Unfortunately the benchmark test software used "Status" in multiple locations as an output parameter. Once this problem was discovered, we modified the software and strange compiler errors were resolved. Other software bugs are described in section 4.2.2.

ORB*express.* Porting the tests to ORB*express* was fairly straightforward. The difficulties encountered were in the C++ client exception handling and the "any" data type extraction on the Server. Also the C++ product that we used does not contain any means of dynamically resolving object references. This caused a change in the benchmark software to specify the server/client endpoints.

TAO. Since TAO and ACE are open source products that only recently became commercially supported, we had to build the ORB and its supporting environment (ACE) for our target platforms (Solaris and LynxOS) as well as port the tests themselves. Our effort to build the ORB was very dependent on the chosen target. For Solaris, the ORB built on the first try and we only made a couple

of tweaks to the configuration based on Washington University's suggestions for performance improvement.

The LynxOS/PPC build was a different story. We spent many hours building the ORB with different options and discussing our build problems with Washington University via email. They were timely and helpful in their responses but we still spent a large amount of time on this process. The benchmark tests themselves ported easily. This was partly due to the fact that the tests had been ported once already and the process was well understood.

4.2.2 Bugs Encountered

HARDPack. Testing using HARDPack was slowed by several problems encountered in HARDPack and additional problems attributed to the Apex compiler. Particularly disconcerting was HARDPack's failure to pass the Basic Data Integrity test. We were also unable to run the IIOP test due to problems in the implementation and the vendor's failure to fix them in the version 1.2 release. The following is a list of Ada95 and C++ problems, workaround and solutions.

Ada95 Problems:

Number 1:

Description: HARDPack1.1 – Bus Error (core dump) when > 75 function invocations
Workaround: Received patch from HARDPack support.
Solution: Ensure that the next version of HARDPack contains the patch.

Number 2:

Description: HARDPack1.1 – Getting "COSNAMING.NAMINGCONTEXT CHILD_UNIT" invalid name exception
Workaround: None required.
Solution: Application of Rational patch 19981119-1.

Number 3:

Description: HARDPack1.1 will not execute on LynxOS
Workaround: None, HARDPack support has written a problem report (144111) with Rational.
Solution: None available

C++ Problems:

Number 1:

Description: HARDPack1.1 idlpp tool generates segmentation faults when arrays are defined as members within a structure. HARDPack problem report 0203
Workaround: Using typedefs, define arrays outside of structures and then use those types as a type in the structure.
Solution: HARDPack STR number PACK0203

Number 2:

Description: The BasicIDL_Client executable is getting a segmentation fault on LynxOS during elaboration.
Workaround: Split the BasicID ORB invocations into three executables.
Solution: None available

Number 3:

Description: The HARDPack1.1 LynxOS includes define “Status” to be void (e.g. “define Status void”.
Workaround: Do not use the word “Status” in source code.
Solution: None available

Number 4:

Description: SPARC/Solaris2.6: When executing a C++ ORB process with real-time priorities the ORB initialization is changing its scheduling mechanisms to time-share.
Workaround: Set the ORB thread priority to 1 in the ORB.init call.
Solution: Document that the C++ ORB.init default threads priority (0) and that this will change the process from real-time to time-share. Note: The Ada95 default is 1.

Number 5:

Description: SPARC/Solaris2.6: When setting the ASE priority in the ORB.init call the following error is be displayed on the console “HARDPack License Has Expired:
Workaround: Change the definition of “*argvt[1]” to “*argvt[2]”
Solution: HARDPack1.1 should give the correct error message.

Number 6:

Description: The data in the first double entry in the “record” data type is not being passed correctly using the Any invocation method. LynxOS only.
Workaround: Don’t use the first double entry in the record
Solution: HARDPack STR number PACK0203

Number 7:

Description: A record is not being passed correctly between machine types (SPARC => PowerPC or PowerPC => SPARC) when transmitted using the “Any” invocation method.
Workaround: None Available
Solution: None Available

Number 8:

Description: SPARC/Solaris 2.6 only. When the Client process is executed with a high “real-time” priority and the Background process is executed with a lower ‘real-time” priority the call from the ORB invocation never returns. Having the Server process (higher “real-time” priority then Background but lower then the Client process) on the same or a different SPARC has not effect. The only way to recover is a power cycle on the Client machine.
Workaround: The Background process is executed with the high “time-share” scheduling policy.
Solution: None Available [Note: problem encountered to some degree with all ORBs tested. Problem to be pursued with Sun Microsystems in cooperation with ORB vendors.]

Number 9:

Description: HARDPack1.1 Client to HARDPack1.1 Server does not work.
Workaround: None Available
Solution: HARDPackSTR PACK0279 is completed.

Number 10:

Description: SPARC/Solaris2.6: The “bc_server” is changing to a non real-time priority. LynxOS: The “bc_server” and “nameserver” is changing to a lower process priority.
Workaround: Command line argument "-p priority”
Solution: Update the HardPack1.1 documentation.

ORB*express.* An early version of ORB*express*'s IDL compiler improperly disallowed the direct extraction of an "Any" type from a server-side actual parameter. OIS fixed this problem in later versions of the IDL compiler. ORB*express* has a different and possibly significant CORBA 2.3 non-compliance problem with "Any" extraction. ORB*express* does not allow the extraction of an "Any" into a pointer type. This method of extraction does not copy the contents of the "Any", but causes the pointer to point into memory controlled by the "Any" data structure. This can be a dangerous coding practice since it strongly couples the use of the "Any" variable and the pointer that points into it. However, this method of extraction can have a significant performance advantage over copying "Any" extraction's for some ORB implementations. Additionally the following is a list of Ada95 and C++ problems, workaround and solutions.

Ada95 Problems:

Number 1:

Description: Large transfer times for the "record" and "not aligned record" transfer methods when using the APEX compiler
Workaround:
Solution: ORB*express* patch 990821864 (APEX array marshalling)

C++ Problems:

Number 1:

Description: Segmentation fault in server when 19216bytes of data is being transfer. ORB*express* PR => ORB990723806
Workaround:
Solution: Increase the server stack size (e.g. CORBA::ORB::stack_size(50000)).

Number 2:

Description: Segmentation fault in server and client when message "delay" set to 70 ms (default for scenario). No segmentation fault when delay set to 0, 10, 20 or 30. ORB*express* PR => ORB991019948
Workaround: Reduce delay to working value of 10, 20 or 30.
Solution: The problem resulted from an undocumented timer associated with use of a demo license in an embedded processor. OIS imposed an absolute execution time limit of 10 minutes on executables built with demo licenses. OIS has been asked to identify this restriction clearly in the release notes distributed with its demo licenses.

TAO. We did not find any problems that could be attributed to errors in the TAO or ACE software.

4.2.3 Summary of Vendor Interactions

Lockheed Martin Federal Systems (LMFS). We began working with LMFS in October, 1998. HARDPack was the first ORB we tested, so some of our difficulties may be attributed to the learning curve required to achieve comfort levels with ORBs in general and the tests we were designing and building. We spent a lot of time interacting with HARDPack support personnel, some of which can be attributed to the learning curve for new HARDPack users. Still, we spent several days of software engineering effort tracking down some of the problems described above.

The responsiveness of the Lockheed-Martin support team varied considerably over the period of our testing. In some instances responses to problems were timely and comprehensive. In others we were forced to contact the support team repeatedly to get answers. We had significant differences with Lockheed-Martin regarding expectations for behavior and capability of a RT ORB. When our expectations did not match, the LMFS response was often an explanation of why a feature or service worked as it did rather than recognition of a potential problem.

Our confidence in HARDPack product support was further eroded when multiple patches and updates failed to resolve the problems they were intended to fix. This was particularly true for our attempts to execute IIOP tests. At least three consecutive releases were supposed to fix IIOP problems but we were never able to use HARDPack's IIOP successfully. Further, in our transition from HARDPack 1.1 to HARDPack 1.2, we found that other working code had been "broken" by the new release.

By mid-August, after expending considerable resources on this product, we terminated testing of HARDPack and focused our attention on achieving comparable levels of testing with ORB*express* and TAO.

Objective Interface Systems (OIS). We started working with ORB*express* in late May, 1999. We started with a trial license for ORB*express* RT for Ada on a Solaris platform and later added ORB*express* GT for C++ for the same platform. In late September we received the Solaris cross-development version of ORB*express* GT targeted to LynxOS for PowerPC. OIS continued to extend the trial licenses so that we could continue our testing. OIS made it very clear that they were interested in getting ORB*express* RT into the RT DII COE. Even though we were not paying customers, we received excellent support from senior management, engineering, sales, and production personnel.

Washington University. We started working with TAO in June, 1999. The Washington University TAO group were very helpful in fixing problems that we found as well as offering advice over email. We were given direct email access to about a half dozen of the top contributors to the TAO project and they responded to our queries in a very timely manner. The commercial support for TAO in the future will come from Object Computing, Inc. (OCI), but we weren't a paying customer and we have not had any communication with OCI except for their responses to the vendor survey.

4.3 Constraints on Development Environment

One of the issues that arose in testing multiple ORBs was that different vendor products required particular operating system and compiler versions. Particularly distressing to both the Boeing RT DII COE team and a user was the fact that HARDPack was built for LynxOS 2.5 and was never upgraded to run on LynxOS 3.0.[9] As a result, organizations that are running multiple COTS products for LynxOS may find it especially difficult to use HARDPack. In addition, the Ada version of HARDPack was only available for Rational Apex, an expensive product that also has numerous problems and frequent patches.

[9] We ran the HARDPack LynxOS 2.5 release on LynxOS 3.0, but the configuration was never officially supported by LMFS.

5. Business Environment Assessments

5.1 Vendor Vital Statistics

We reviewed information related to potential longevity of the RT CORBA product. This is important because the expected life of DOD systems is often 20 years or more. As we have noted before, the RT CORBA vendors included in this span a wide range of organization types: Lockheed Martin Federal Systems (LMFS), is a defense prime contractor, Objective Interface Systems, Inc. is a small commercial software vendor specializing in CORBA products, Objective Computing, Inc. is a small software engineering firm, and Washington University is an academic institution. We did not attempt to obtain financial information that is not publicly available.

5.1.1 Lockheed Martin Federal Systems - HARDPack

Address: 1801 Star Route 17C, Owego, New York 13827-3998

Point of Contact: Patrick Morrissey, AWACS Systems Engineering Manager, 607-751-3811

Overview. "Lockheed Martin is one of the world's leading diversified technology companies. We research, design, develop, manufacture and integrate, advanced technology systems, products, and services for government and commercial customers around the world. Core businesses span aeronautics, space, systems integration, and technology services."[10]

- First RT CORBA product released: October, 1998
  (HARDPack - its predecessor, DSI, was first released in 1996.)
- Product Development Funding Sources:
  67% AWACS Modernization Program
  33% Lockheed Martin.
- Publicly Traded Corporation
- 1998 Net Sales: $26.26 billion (All of Lockheed Martin)
- Number of Employees: >170,000 (All of Lockheed Martin)
- Percentage of Employees dedicated to implementation of CORBA products: much less than 1%
- Percentage of Employees dedicated to customer support for CORBA products: much less than 1%
- Percentage of Employees that perform both implementation and customer support: much less than 1%

Discussion: While Lockheed Martin is a large corporation with significant resources, RT CORBA is a minor product line. The focus in product development appears to be in meeting the needs of the AWACS program, which was the major source of funding for HARDPack. This is

[10] Copied from http://www.lmco.com/ .

evident in the availability and plans for product features required by AWACS but not those specified in the RT CORBA standard. In a telephone conference with Boeing on July 14, 1999, Lockheed Martin Federal Systems (LMFS) stated that its primary focus for HARDPack is to embed it in total solutions, not sell it as a "shrink-wrapped" product. Finally, in a November 12, 1999 telephone conference with Boeing, LMFS stated that while it would continue to support existing HARDPack customers, it would refer future inquiries regarding CORBA products to Objective Interface Systems.

5.1.2 Objective Interface Systems - ORB*express*

Address: 1892 Preston White Drive, Reston, VA 20191-5448

Point of Contact: Steve Grimaldi, Director of Engineering, 703-295-6522

Overview. " Objective Interface provides software developers tools for building reliable, mission-critical systems. These tools include CORBA compliant Object Request Brokers, training and support to implement software systems in a cost-effective and reliable manner. Our primary focus and expertise also includes high performance solutions for complete Real-Time and embedded applications."[11]

- First RT CORBA product released: 1995 - Orbix Ada
- Product Development Funding Sources: 100% Objective Interface Systems
- Private Company founded in 1989.
- Number of Employees: 20
- Percentage of Employees dedicated to implementation of CORBA products: 50%
- Percentage of Employees dedicated to customer support for CORBA products: 25%
- Percentage of Employees that perform both implementation and customer support: 0%

Discussion: Development of real-time CORBA products is the major focus for OIS. While the company is small, it appears to have a broad, growing user base. Boeing team experience and comments received in the user survey generally reflect high satisfaction with the ORB*express* products and customer service. OIS maintains an extensive, informative website that is updated frequently. Based on all of the information available to us and interactions with company personnel, we believe the prospects for long-term support for ORB*express* products are good.

[11] Copied from www.ois.com

5.1.3 Object Computing Inc. (OCI) / Washington University- The ACE ORB (TAO)

Address: 12140 Woodcrest Drive Suite 250 (OCI address)
St. Louis, MO 63141

Point of Contact: Malcolm Spence, Director of Business Development, (314)-579-0066

Overview. "Object Computing, Inc. is a Software Engineering company dedicated to providing innovative solutions using Object-Oriented and web-enabled technologies."[12] The primary goal of the Center for Distributed Object Computing at Washington University is to "support advanced R&D on distributed object computing middleware using an open source software development model."[13]

- First RT CORBA product released: July, 1999
- Product Development Funding Sources: Defense Advanced Research Projects Agency (DARPA), National Science Foundation, USENIX and industry. Industry contributors include BBN, Bellcore/Telcordia, Boeing, Experian, Garg Data International (GDIS), Global Main Tech, Kodak, Krones, Lockheed, Lucent, Microsoft, Motorola, Nokia, Nortel, OCI, OTI/IBM, SAIC, Siemens, and Sprint.

	OCI	Washington University (TAO project only)
• Number of Employees	40	17
• Percentage of employees dedicated to implementation of CORBA products	30%	100%
• Percentage of employees dedicated to customer support for CORBA products	20%	0%
• Percentage of employees that perform both implementation and customer support	20%	0%

Discussion: Dr. Douglas Schmidt of Washington University (WU) has been the driving force behind TAO. At the end of 1999, Dr. Schmidt and a small part of the Center for Distributed Object Computing (DOC) staff will be moving to the University of California at Irvine (UCI) and Dr. David Levine will take over leadership of the Center for Distributed Object Computing at WU. UCI and WU will work together on a variety of research projects, including TAO and ACE. In late spring of 2000, Dr. Schmidt will assume a position at DARPA where he will be responsible for research related to adaptive real-time middleware for embedded systems. His goal is to raise the level of respect for middleware and to take RT CORBA to the next level of maturity. While at DARPA, he will continue to be involved with various aspects of the R&D efforts at UCI and WU.

OCI has an agreement with WU to make the technology available with comprehensive commercial support. Under this agreement, which was announced in December, 1998 (reference 6), OCI performs testing, marketing, training, and support functions. Since the agreement was put in place, OCI has made significant strides in commercializing TAO including generating a

[12] Copied from www.ociweb.com
[13] Copied from http://www.cs.wustl.edu/~schmidt/doc-center.html

documentation set and enabling on-line ordering. OCI intends to open an office near UCI early next year.

WU and OCI both maintain extensive, informative websites for TAO that are updated frequently. The fact that the product is free and source code is available is attractive and the number of users appears to be growing. In addition, there is a large worldwide community of software developers that have contributed to TAO and are familiar with its construction. Other examples of open source products supported by commercial ventures exist, and commercialization of TAO is being patterned after the more successful examples. While it is too soon to judge how successful the commercialization will be, at this point is seems reasonable to believe that TAO will be available and supported for the long term.

5.2 Comparative Pricing Data

For completeness, we collected pricing information for each of the three products and compared them for a few different development scenarios. While the comparison may be interesting, it is important to remember that the price of the RT CORBA product must be placed in the context of the program's entire life cycle cost. In most cases the cost of the product licenses and support will not be a major factor in a decision as to which product to use.

As shown in Table 5-1, there is considerable variation in the pricing approaches used by the three product vendors. They each have different policies as to the types and numbers of licenses that must be purchased and the cost of support for those licenses. Pricing data shown in this document is for rough comparative purposes only and should not be used in lieu of actual vendor quotes.

Table 5-1. CORBA Product Pricing Overview

	HARDPack	**ORB*express***	**TAO**
Developer License Fee	Flat fee - one license per user covers Ada and C++ for multiple platforms	Licensed by developer name, one license per platform & language	• None • Software is free if downloaded or $60 for CD-ROM • $350 for OCI documentation set
Developer License Discounts	• DOD • Quantities of 5 or more	• Quantity breaks: 3,5, 10 • Multi-packs – combinations of platforms and/or languages	Not applicable
Run time License	• One time flat fee per license • One license included with developer license • DOD discount • Quantity discounts	None	• None
Maintenance Fee	Yearly fee per developer license	Yearly fee per developer license; 1st year included in developer license	• Flat fee by support level; allows for three points of contact (eff. 1Q 2000) • $125/hr payable by credit card (current)
Source Code License	Fee variable and negotiable. Option to hold source code in escrow also available.	Fee variable and negotiable. Option to hold source code in escrow also available.	No charge No restrictions
Vendor Site Training	$1500 for 5-day class in Owego, NY	$1495 for 3-day ORB*express* for C++ in Reston, VA $1995 for 5-day ORB*express* for Ada in Reston, VA	$1400 for 4 day class in St. Louis, MO or Phoenix, AZ 3 different classes available
Customer Site Training	$25,000 + travel for 5-day class	$16,000 for 5-day class	$14,000 + expenses for up to 16 students

We have not included actual dollar values in the table above because prices may change rapidly and the ORB*express* pricing model is complicated and proprietary. Approximate prices for the three ORB products are shown graphically in the pricing scenarios below.

We have chosen five examples to illustrate how changes in development scenario affect the relative prices. The first four examples show cumulative product pricing comparisons for varying numbers of developers, platforms (processor/operating system), and programming languages. In all cases the following assumptions apply: (1) 5 run-time licenses, regardless of the number of developers, (2) constant 1999 dollars, (3) constant number of developers throughout the five year project life, and (4) constant maintenance fees over the 5-year comparison period. The fifth example outlines a slightly more complex scenario for a hypothetical program.

Figure 5-1 shows the simplest case, a single developer on a single platform using a single programming language. As shown, in the front end of a program, ORB*express* is the most expensive. Over time, it becomes the least expensive because it has a significantly lower per year maintenance fee. TAO is the least expensive up front but single price maintenance plan is expensive since it assumes at least three developers.

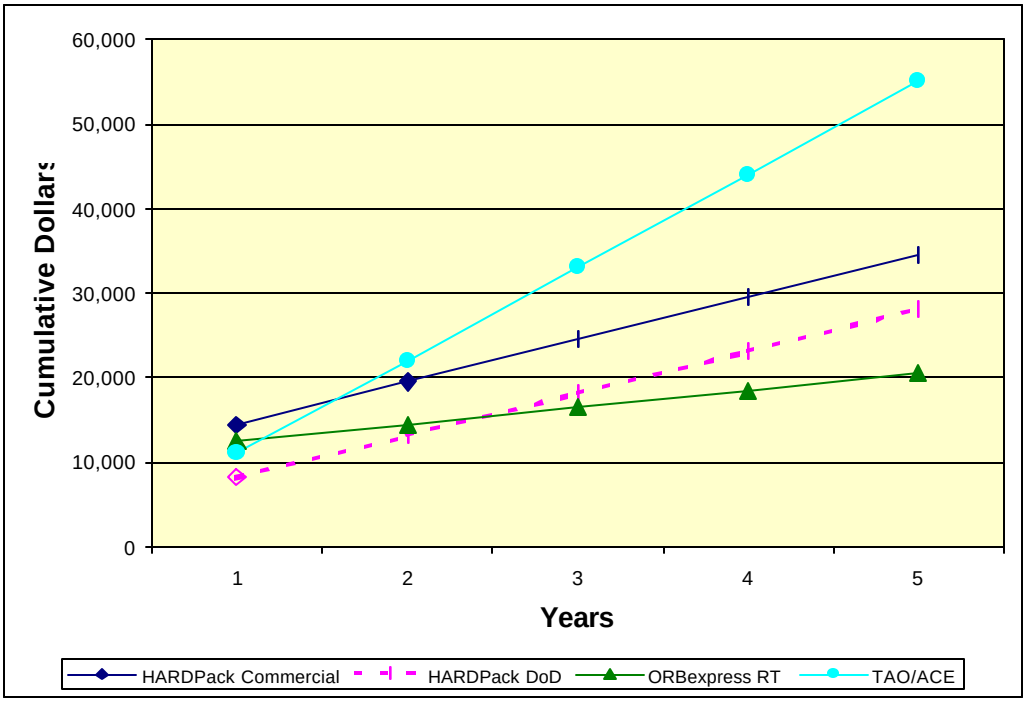


Figure 5-1. Pricing Comparison: 1 Developer, 1 Platform, 1 Language

Figure 5-2 comparison for three developers using a single platform and language, shows the advantage switching to TAO since the price had already assumed three points of contact. This advantage grows even more significant as the number of developers increases as long as the user organization can organize itself to route all interactions with OCI through three people.

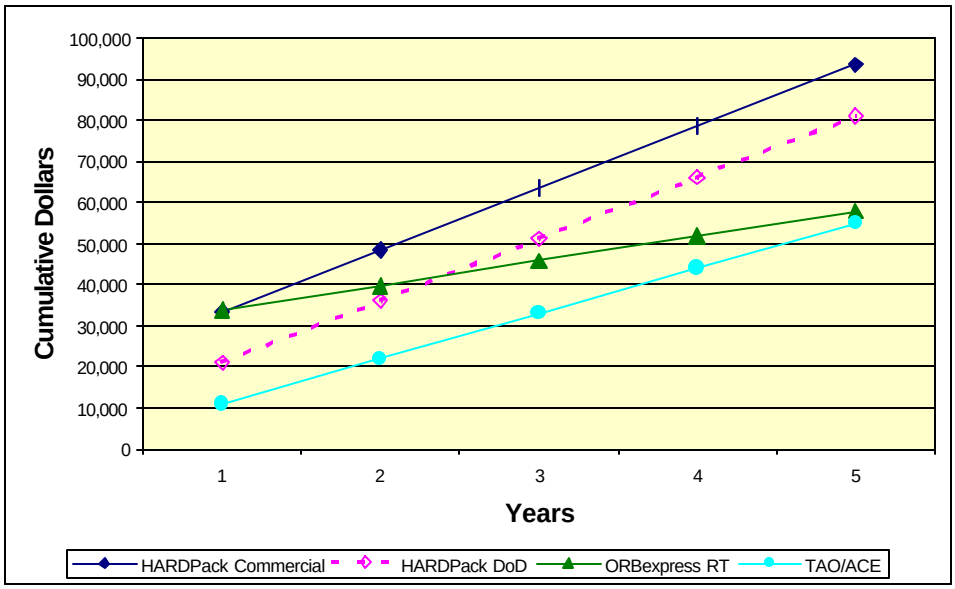


Figure 5-2. Pricing Comparison: 3 Developers, 1 Platform, 1 Language

In Figure 5-3 we show the effects of adding a second platform, in this case identified as a cross-linked target to the single developer, single language case. ORB*express* is less attractive than in the single platform case because of the additional developer license fee and higher yearly maintenance price.

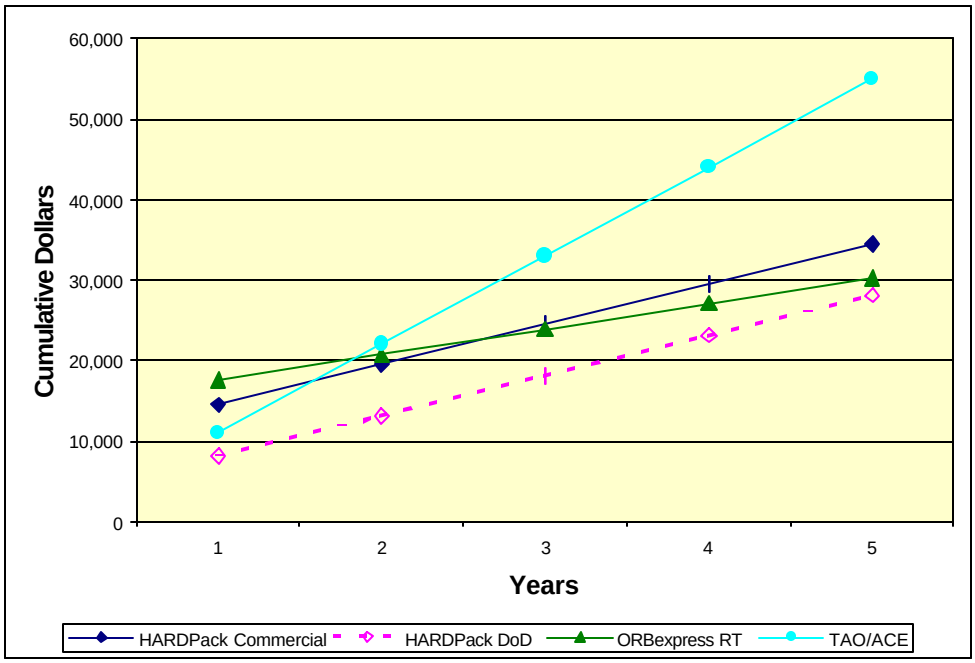


Figure 5-3. Pricing Comparison: 1 Developer, 2 Platforms, 1 Language

In Figure 5-4, where five developers each require use of Ada and C++ versions for self-hosted and cross-linked platforms, the ORB*express* approach of pricing by platform, by language, by developer puts the product at a serious disadvantage relative to HARDPack. TAO is not included in this comparison since Ada versions are not available.

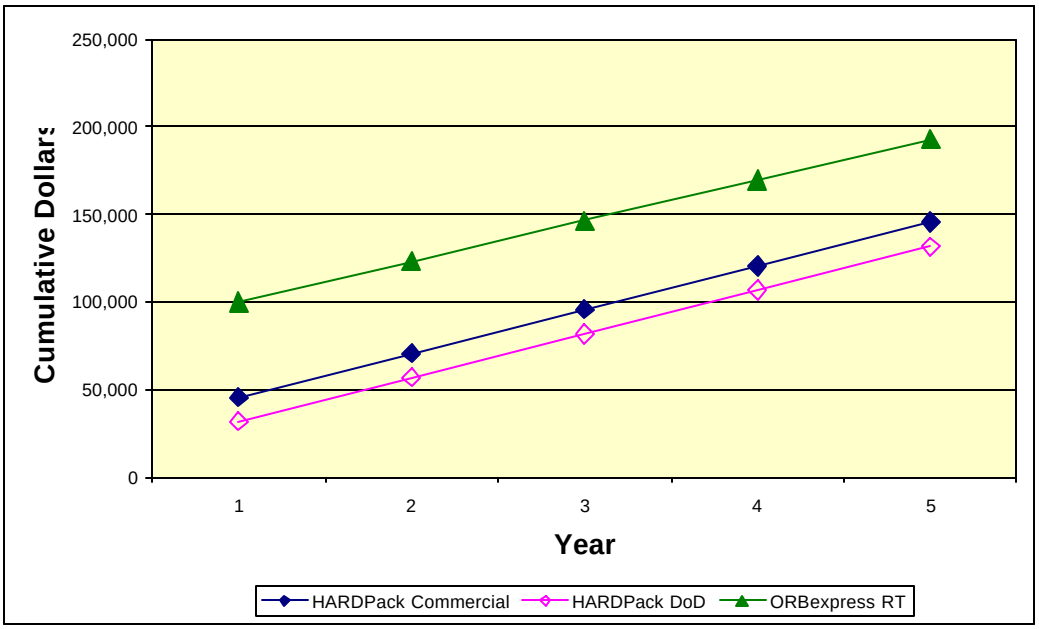


Figure 5-4. Pricing Comparison: 5 Developers, 2 Platforms, 2 Languages

In the final example, we have a hypothetical development program requiring 10 developers to use the RT CORBA product. They each need to develop software using IDL bindings for C++ only and will develop on a Sun/Solaris system for execution on a PowerPC running a real-time operating system. The prices of the development and target environments are not included in this calculation. All procurement of CORBA products is done by a contractor, not a Government agency.

Ten developer licenses, with 10 run-time licenses included, will be purchased. Yearly maintenance is purchased for each, but after the third year only three development environments will continue to be maintained. This maintenance requirement will extend throughout the life of the system. In the fifth year the system will be deployed to 200 CPUs, requiring 193 additional run-time licenses. The expected life of the deployed system is 10 years. Figure 5-5 shows the cumulative costs over the life of the hypothetical program.

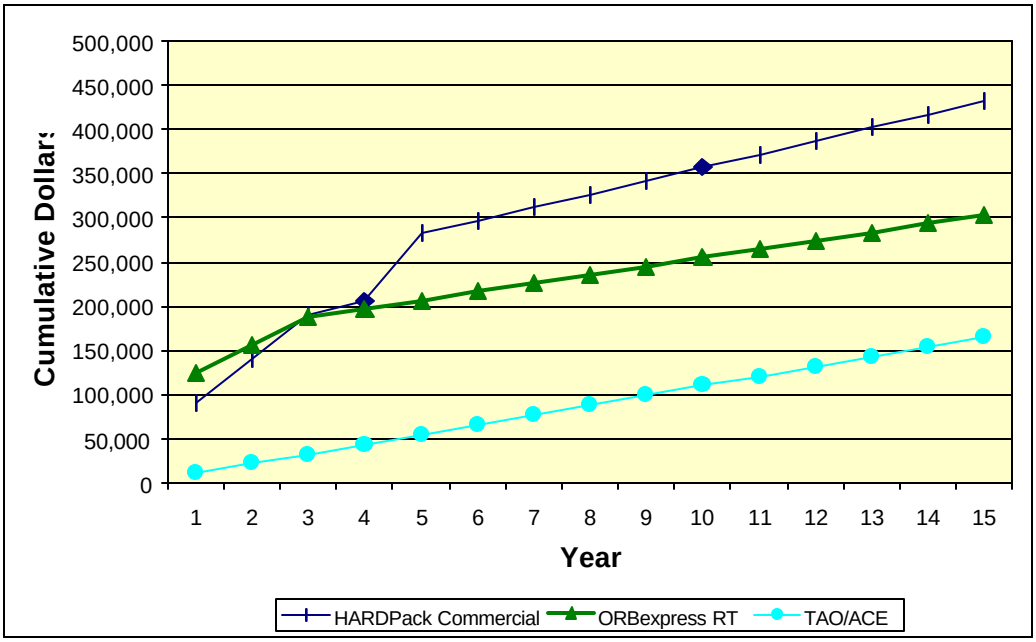


Figure 5-5. Pricing Comparison: Hypothetical Program

5.3 Expectations for Product Sponsorship

For nomination into the COE, a vendor product must have two Government program sponsors. Table 5-2 shows the expected sponsors as of October 15, 1999.

Table 5-2. Potential Product Sponsors

HARDPack	ORB*express*	TAO
NATO Airborne Warning and Control System (AWACS)	Advanced Field Artillery Tactical Data System (AFATDS)	Joint Tactical Terminal (JTT)
Regional/Sector Air Operations Center (R/SAOC)	Joint Tactical Radio System (JTRS)	Upgraded Early Warning Radar (UEWR)

6. Conclusions and Recommendations

While the RT DII COE IPT has completed its RT CORBA evaluation, there is still much to learn. We measured the basic performance of three ORBs at the level of single operations but barely scratched the surface on their abilities to handle real-time requirements. We did some initial interoperability testing with good results. We will be able to explore additional issues in this area as we move into integration testing of the DII COE RT extensions.

During the evaluation, we have coordinated with other organizations performing ORB benchmarks. These organizations help fill holes in areas that we have not tested and render a second opinion in areas that we have. We have compared our results with those obtained by the Upgraded Early Warning Radar (UEWR) program and look forward to obtaining results from the Naval Surface Warfare Center (NSWC). The IPT will continue to act as a clearinghouse for other organizations wishing to share information.

Potential users of the results contained in this report must recognize that at best, this work provides an initial filter. The choice of product for any particular program still depends on program specific factors such as environment, application, system architecture, and product support requirements. We recommend that our results be augmented with system specific testing before final decisions are made. This disclaimer aside, we make the following recommendations:

- For systems being developed in Ada, ORB*express* RT is a viable candidate.
- For systems using C++, ORB*express* GT (RT when available) and TAO are both viable candidates. Both are available for multiple compilers on a wide variety of platforms and have demonstrated that new ports can be added relatively easily. Our testing indicates that both deliver good basic performance and capability, but there are subtle differences. ORB*express* GT was better in terms of raw performance, while TAO offers more services and more options for ORB configuration. (TAO has more configuration parameters exposed to the user and available for tuning and tailoring of ORB behavior.) The experience and personality of the development and management teams should influence the selection. They will need to consider whether they would be more comfortable with a carefully packaged commercial product or an open source product. Finally, TAO has an obvious advantage in licensing cost, although the implied costs of operating in an open source environment should not be ignored.

Next Steps. Based on the concurrence received from the RT TWG November 4, 1999, the IPT will work with both Washington University/OCI and Objective Interface Systems to prepare to nominate both TAO and ORB*express* RT into the DII COE. The IPT will also continue to monitor and report on major changes in the RT CORBA market and on the results of other benchmarking activities. The success of this activity will be significantly affected by voluntary contributions on the part of IPT and TWG participants. In addition, the IPT will monitor the evolution of products to RT CORBA 1.0 and will perform some assessments of real-time behavior as we perform integration testing.

Finally, we would like to thank the users, vendor personnel, and IPT sponsors that helped us with this evaluation. Without their input, this effort could not have been accomplished.

Glossary

ACE	ADAPTIVE Communication Environment
ADAPTIVE	A Dynamically Assembled Protocol, Transformation and Validation Environment
AWACS	Airborne Warning and Control System
BDI	Basic data integrity
CORBA	Common Object Request Broker Architecture
CPU	Central processor unit
CR	Call & Return
DARPA	Defense Advanced Research Project Agency
DII COE	Defense Information Infrastructure Common Operating Environment
DISA	Defense Information Systems Agency
DoD	Department of Defense
IDL	Interface definition language
IIOP	Internet inter-ORB protocol
IPT	Integrated Product Team
JTT	Joint Tactical Terminal
LMFS	Lockheed Martin Federal Systems (Produces and supports HARDPack)
OCI	Object Computing, Inc. (Supports TAO)
OIS	Objective Interface Systems (Produces and supports ORB*express*)
OMG	Object Management Group
ORB	Object request broker
OS	Operating system
OW	One-Way
POA	Portable Object Adapter
PPC	Power PC
RT	Real-time
RTOS	Real-time operating system
TAO	The ACE ORB
TWG	Technical Working Group
USAF	United States Air Force
UCI	University of California at Irvine
WU	Washington University

References

1. Defense Information Systems Agency, “Defense Information Infrastructure (DII) Common Operating Environment (COE) Integration and Runtime Specification (I&RTS), Version 4.0.” October 1999.

2. Lt. Col. Lucie M.J. Robillard, Dr. H. Rebecca Callison, and John Maurer, “Extending DII COE for Real-time,” Crosstalk, The Journal of Defense Software Engineering. Volume 12, Number 9, p. 6-12, September 1999.

3. Michele Evans, Senior Program Manager, Lockheed Martin Federal Systems Owego, to Mr. Fletcher Thomson, Senior NAPMA Representative, NATO Airborne Early Warning Programme Management Agency, ESC/AW, “Committed HARDPack Development Plan.” July 12, 1999.

4. README file included with HARDPack 1.3 release dated October 15, 1999.

5. orbos/99-02-12, “Real-Time CORBA Joint Revised Submission,” Object Management Group (OMG) Technical Committee (TC), March 1, 1999.

6. Object Computing Inc. Press Release, “The ACE ORB’ (TAO) Obtains Commercial Support”, dated December 3, 1998: http://www.ociweb.com/What_s_New/Press_Releases/tao_released.html.

Acknowledgements

We would like to thank the many people and organizations that assisted us with the study.

Boeing engineers Gary Noel, Dan Butler, Rob Olsen, and Jim Higashi who spent many early mornings, late nights, and weekends helping Becky Callison produce and analyze the benchmark test data.

Personnel from Lockheed Martin Federal Systems, Objective Interface Systems, Object Computing, Inc. who responded to the vendor survey.

Expert reviewers from the Naval Surface Warfare Center, Joint Tactical Terminal Program, and MITRE-Bedford who assisted in analyzing the vendor survey results.

The ORB*express* and TAO engineering and support teams that went far beyond the call of duty in providing technical assistance to the Boeing engineering team.

All users who responded to the survey.

People who reviewed the draft: Bill Beckwith and Steve Grimaldi of Objective Interface Systems, Doug Schmidt of Washington University, Malcolm Spence of Object Computing, Inc., Dave Emery and John Maurer of The MITRE Corporation, Lt. Col. Lucie Robillard, Joe Gwinn of Raytheon, and Hal Hawley, Steve Uczekaj, Christine Doan, and Dave Munro of The Boeing Company.

Appendix A Vendor Questionnaires

DII COE Real-time Technical Working Group (RT TWG)
ORB Vendor Technical Questionnaire

The RT TWG has established a number of requirements it desires ORBs and other products contributing to the DII COE real-time distributed computing infrastructure to satisfy. We provide you with the title, identifiers, and text for each of these requirements.

We request that you assess your own product(s) and advise us of the degree to which you believe they satisfy each TWG requirement. For each requirement, we have provided space to present a rationale, substantiating your claim of satisfaction. An additional response field provides the opportunity to describe the techniques you suggest for verifying your claim.

Your responses will be used by the RT TWG as an aid in prioritizing which ORB product(s) to initially evaluate and ultimately recommend as part of the initial DII COE RT computing infrastructure.

It is not expected that all products will satisfy all requirements of the RT TWG. Please contact Michael Grieco (mjgrieco@jswg.org, (301) 483-6000 ext. 2451), if you need clarification of these requirements. Responses should still be returned to Rebecca Callison (rebecca.callison@boeing.com, (253) 657-3952).

Please describe the way your orb product(s) satisfy each of the following requirements established by the RT TWG for real-time distributed computing infrastructure:

Title: Bounded blocking time in ORB operations:

Requirement ref ID	AIRDC0048
Text	Is ORB designed so that any other processes do not block ORB operations (e.g. method invocations and interfaces to these mechanisms) indefinitely due to the obvious determinism problem it would cause. In other words, ORB blocking time must be bounded and specified.
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Title: Replaceable transport protocol:

Requirement ref ID	AIRDC0049
Text	Does the ORB support transport protocols other than TCP (Please describe) *Amplifying Note:* The RT CORBA 1.0 specifies that multiple transports can be selected through the *ProtocolPolicy* and *ProtocolProperties* features, however these must be "implementation specific" by default. The OMG Pluggable Transport RFP (orbos/98-07-16) was an initiative to create an interface that facilitates introducing a real-time transport layer into the architecture however, it appears that the initiative may have stalled.

Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Title: Priority-based queuing:

Requirement ref ID	AIRDC0046
Text	Do client requests to the server to have an associated priority so that the server can respond to the client request accordingly? Is Queuing FIFO within a single priority?
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Title: Global recognition and communication of priorities:

Requirement ref ID	AIRDC0045
Text	Includes priority migration. Priority scheme must address both the transport of messages and priorities of the server tasks.
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Title: "End to end" deadline guarantee:

Requirement ref ID	AIRDC0050
Text	Does the ORB have features that contribute to the desired *system* level deterministic behavior in addition to the requirements listed in AIRDC0048, AIRDC0049, AIRDC0046, and AIRDC0045?
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	

Title: Support bypassing marshal / demarshal process:

Requirement ref ID	AIRDC0051
Text	Do you support bypassing the marshal/de-marshaling of data when communication is between objects that are coded in the same language and running in the same process.

Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Title: Interface to transport layer QoS:

Requirement ref ID	AIRDC0031
Text	Does ORB provide the ability to use Quality of Service (QoS) capabilities (e.g., RSVP, Differentiated Services, ATM QoS, IEEE 802.1p) to provide Guaranteed Quality of Service as defined in IETF RFC 2212?
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Title: Support asynchronous message-based system architectures:

Requirement ref ID	AIRDC0032
Text	Does ORB provide support for the asynchronous message-based system architectures common in sensors and weapons?
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Title: Mechanism for avoiding priority inversion:

Requirement ref ID	AIRDC0047
Text	Does the ORB provide mechanisms to avoid priority inversion in ORB operations? Please describe what method is used
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

The following additional requirements have been proposed and are under discussion. Please comment on the degree to which your product satisfies these requirements as well. Use the same format for reporting as for the current requirements listed above.

Support reliable and unreliable unicast and multicast:

Requirement ref ID	-
Text	Typical systems require support of both reliable and unreliable unicast and multicast (all four combinations), implemented such that end to end latency, latency jitter, and CPU cost at endpoints scales no worse than linearly with both endpoint load and system-wide traffic. i.e. the CORBA event and notification services.
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Small Memory footprint for Core ORB and Services:

Requirement ref ID	-
Text	The memory footprint for the Core ORB and associated CORBA services needs to be a small as possible in order to be contained in embedded systems.
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comment	

Persistent Bindings:

Requirement ref ID	-
Text	Does the ORB provide support for persistent bindings; bindings that persist between executions of an application? (This requirement is derived from performance and predictability needs. Can be used if objects are not expected to move to different locations during system operation.
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Scalability

Requirement ref ID	-
Text	Comment on the scalability in configurations and in performance characteristics.
Degree to which requirement is satisfied	
Rationale for claim of satisfaction:	
Mechanism proposed for verifying satisfaction:	
Comments	

Please complete the following list regarding service, platform, and language support provided by your ORB.

Capability/Service	Vendor Description of Support
IIOP	
IDL compiler	
IDL Bindings:	
ANSI C C++ Ada95 Java	
Static invocation interface	
Naming service	
Event Notification services	
Time service	
Implementation repository	
Interface repository	
Trader service	
Dynamic invocation interface	
Transaction service	
Other Services of Possible Interest	

DII COE Real-time Technical Working Group (RT TWG)
ORB Vendor Non-Technical Issues Questionnaire

At the request of the Real-time Defense Information Infrastructure Common Operating Environment (RT DII COE) Technical Working Group (TWG), the RT DII COE Integrated Product Team is evaluating CORBA products for suitability for inclusion in the DII COE. As part of this evaluation, we need to consider vendor commitment to long term product support, portability across a variety of real-time platforms, and interest in having the product included in the DII COE. Other aspects of the evaluation include testing of functionality and performance, and conformance with the RT CORBA standards.

To assist us in our evaluation, please answer the following questions:

CORBA Product Developer: _____

Address: _____

City, St. Zip Code _____

Point of Contact: _____

Telephone: _____ Email:__ _____

1. Product Line
   a. What do you consider to be your target markets?
   b. What are your organization's top three products?
   c. Which, if any, of your products have been segmented in accordance with the DII COE Integration and Run-Time Specification (IRTS)?
   d. Which, if any, of your products do you expect to segment for the Real-time extensions to the DII COE that are currently being developed?
   e. Please provide pricing information for your real-time CORBA products:

      (1) Developer license_____

      (2) Run time license _____

      (3) Maintenance _____

      (4) Other (describe volume discounts, etc.)__

2. Product History
   a. When was your first CORBA product released?
   b. Who funded development of your real-time CORBA product? (Please identify all organizations that provided significant funding and the approximate percentage of the total funding that each organization contributed.)
   c. For each release of your real-time CORBA product, please provide the following:

Version	Release Date	Applicable CORBA Standard	Technical Highlights

3. Real-time CORBA Compliance
   d. Does your product comply with CORBA 2.2?

      (1) If your product is not fully compliant with CORBA 2.2, what features are not currently implemented?

      (2) What is the most recent CORBA revision with which your CORBA products are fully compliant?

      (3) When do you plan full compliance?

   e. Does your product comply with CORBA RT 1.0?

      (4) If your product is not fully compliant with CORBA RT 1.0, what features are not currently implemented? (Please use the companion conformance questionnaire, supplied with this document, to indicate areas of conformance.)

      (5) When do you plan full compliance?

   f. What methods do you use to verify your claims of compliance?

      (6) For example, has compliance of your product been successfully tested by an independent testing organization?

      (7) If independent compliance testing has been completed, by whom and when?

      (8) If no such testing has been accomplished, do you plan such testing in the future?

4. User Base
   - Please indicate the approximate number of current (non-expired) **run-time** licenses for your CORBA products that are being used in fielded (end user) systems:
     Commercial____ Government____ Education_____
   - Please indicate the approximate number of current **run-time** licenses for your CORBA products have been sold for use in systems currently in development:
     Commercial____ Government____ Education_____

- Please indicate the approximate number of **development system** licenses that have been sold to:
- Commercial____ Government____ Education_____

5. User Contacts

We would like to interview users of CORBA products. Please provide point of contact information (organization, POC name, telephone number, and email address) for up to ten (10) distinct organizations that are currently using your CORBA product(s):

a. As part of fielded real-time systems (end users).

b. As part of fielded non-real-time systems.

c. To develop real-time systems

d. To develop non real-time systems

6. Portability

g. Please indicate which of the following your CORBA products currently support. If the platform is not currently supported but is scheduled for release, please indicate when the release is planned.

h. How many weeks after release of a major (new features, not just bug fixes) operating system upgrade does it take for you to release a compatible upgrade to your CORBA products?

i. Approximately how long would it take to port your product to a new operating system if a customer ordered a version of your product using a language that you already support and a hardware platform that you already support.

j. Would the customer be asked to pay the non-recurring costs?

7. Product Stability and Support

Many of the programs in our community expect to have a 3-year (or longer) development cycle followed by a minimum of 10 years in the field. The following questions are included to evaluate supportability of systems that include your product.

k. What documentation do you provide with your RT CORBA products?

l. Please provide a copy of your latest release memo or release notes for your RT CORBA product(s).

m. Please describe what training is available for users of your product, who provides it, and how much does it cost?

n. What organization is responsible for fielding of customer technical questions?

o. Are customer problem reports accepted via:

1. Email ____
2. Web site ___

3. Telephone ___
4. Other ____ _____

p. What is the average response time following receipt of a customer problem report?

q. How often are problem resolutions incorporated and released?

r. What mechanisms are used to inform the general user community of problems found and resolved between these releases?

s. Briefly describe the process used to test your CORBA products (or attach a relevant document.)

t. Briefly describe the process used to manage the configuration of your CORBA products (or attach a relevant document.)

u. Briefly describe the process used for quality assurance of your CORBA products (or attach a relevant document.)

v. Does purchased maintenance include product upgrades?

8. Restrictions and Warranties

w. What, if any, caveats, restrictions or disclaimers do you stipulate regarding the use of your CORBA product(s) and related services in mission critical and/or safety critical applications?

x. Briefly describe any warranty provided with your RT CORBA products.

9. Other

y. How many employees does your organization have?

z. What percentage of the employees are dedicated to implementation of CORBA products?

aa. What percentage of the employees are dedicated to customer support for CORBA products?

bb. What percentage of the employees perform both implementation and customer support functions?

DII COE Real-time Working Group

Conformance Statement Questionnaire

for the

CORBA - Real-time Extensions

7 July 1999

Prepared by

Defense Information Systems Agency

Joint Interoperability and Engineering Organization

Center for Standards

10701 Parkridge Blvd

Reston, VA 20191

1 Introduction

This document is intended to be a guideline for the Defense Information Infrastructure Common Operating Environment (DII COE) Conformance Statement Questionnaire (CSQ) dealing with real-time systems.

2 Conformance Statement Questionnaire

This document follows the major sections of the Real-time CORBA specification (OMG TC Document orbos/99-02-12).

Global Assumption. CORBA 2.2 specification is expected as the default.

2.1 Real-time ORB

Feature	Yes	No
Has the *RT_CORBA::RTORB* interface been implemented?		
If yes, is it locality constrained?		
Does the ORB_init operaton initialize all the Realtime capabilities of the ORB? (assuming that CORBA compliance is a given)		
If yes, does it support entering an RT ORB priority range?		
Have the additional *Standard Minor Exception Codes* for Real-time CORBA been implemented? Including:		
MARSHAL?		
DATA_CONVERSION?		
INITIALIZE?		
BAD_INV_ORDER?		
NO_RESOURCES?		

2.2 Real-time POA

Feature	Yes	No
Has the *RTPortableServer::POA* interface been implemented?:		
If yes, does it support all POA semantics?		
If also yes, does it support object level priorities?		
If both yes, does it have support for RT policies?		

2.3 Native Thread Priorities

Feature	Yes	No
Have *Native Thread Priorities* been implemented? Including:		
Native thread priority scheme?		
Base native thread priority?		
Active native thread priority?		
Has *Priority Inheritance* been implemented? Including:		
Priority inheritance protocol?		

2.4 CORBA Priority

Feature	Yes	No
Has the *RT_CORBA::Priority* type been implemented?		

2.5 CORBA Priority Mappings

Feature	Yes	No
Has the *PriorityMapping* native been implemented? Including:		
C bindings?		
C++ bindings?		
ADA bindings?		
JAVA bindings?		

2.6 Real-time Current

Feature	Yes	No
Has the *RT_CORBA::Current* interface, been implemented?		

2.7 Real-time CORBA Priority Models

Feature	Yes	No
Has the *PriorityModelPolicy* interface been implemented? To include:		
Client Propagated Priority Model (CLIENT_PROPAGATED)?		
Server Declared Priority Model (SERVER_DECLARED)?		

2.8 Priority Transforms

Feature	Yes	No
Has the *RT_CORBA::PriorityTransform* interface type been implemented? To include:		
C bindings?		
C++ bindings?		
ADA bindings?		
JAVA bindings?		

2.9 Mutex Interface

Feature	Yes	No
Has the *RT_CORBA::Mutex* interface been implemented?		

2.10Thread Pools

Feature	Yes	No
Have *ThreadPools* been implemented? To include:		
create_threadpool?		
create_threadpool_with_lanes?		
Request buffering?		

2.11Implicit and Explicit Binding

Feature	Yes	No
Has the *CORBA::Object::validate_connection* operation been implemented? To include:		
Implicit binding (default CORBA behavior)?		
Explicit binding?		

2.12Priority Banded Connections

Feature	Yes	No
Has the *PriorityBandedConnectionsPolicy* policy type been implemented? To include:		
RT_CORBA::PriorityBand?		
RT_CORBA::PriorityBands?		
RT_CORBA::Priority values?		
RTCorbaPriorityRange?		

2.13Private Connection Policy

Feature	Yes	No
Has the *PrivateConnectionPolic*y interface been implemented? To include:		
PRIVATE_CONNECTION_POLICY_TYPE?		
create_private_connection_policy?		

2.14Invocation Timeout

Feature	Yes	No
Has the *Messaging::RelativeRoundtripTimeoutPolicy* been implemented?		

2.15Protocol Configuration

Feature	Yes	No
Has the *ServerProtocolPolicy* been implemented? To include:		
RT_CORBA::ProtocolProperties?		
Has the *Client**ProtocolPolicy* been implemented? To include:		
RT_CORBA::Protocol?		
RT_CORBA::ProtocolList?		
IOP::ProfileId?		

2.16Real-time CORBA Scheduling Service

Feature	Yes	No
Has the *Real-time CORBA Scheduling Service* been implemented?		

Appendix B User Questionnaire

Product Identification:

Product Name:
Product Vendor:

1. Identification of Responder:

Company Name:
Organization Name
Name of Responder:
Position of Responder:
Contact info for Responder:
Email:
Phone
Fax
Postal address

2. Types of application system(s) in which ORB product is or has been used: (check all that apply)

(a) Timing Constraints: Hard real-time_____ Soft real-time ____

(b) Environment: Embedded (not workstation) _____ Workstation _____

(c) Application Types: (e.g., Airborne Surveillance and Warning)_____

(d) _____

(e) _____

3. Describe releases of product and relevant execution environments over entire history of use, if possible.

History of Product Use				
ORB / Release #[14]	Execution Environment			Approximate Dates of Use
	CPU type	OS (by release #)	Compiler (language & vendor)	

[14] List as many releases as are relevant to this survey.

COMPLEXITY

4. For your most complex application:

(a) Please fill in the table below related to software objects connected by the ORB.

CPU	Operating System	No. of Objects

(b) Describe the typical complexity of an object interface? (Number of methods, number of parameters per method)

ORB ATTRIBUTES

5. What features of the [RT] ORB are you actually using in your [real-time] application? Please rate where "1" is most critical to program success and "5" is least critical to program success.

Capability/Service	Rating	Capability/Service	Rating
IDL compiler		RT Event Notification services	
IDL Bindings:		Time service	
ANSI C		Implementation repository	
C++		Interface repository	
Ada95		RT Trader service	
Java		RT Transaction service	
Other _____		Naming Service	
IIOP		Messaging service	
Static invocation interface		Asynchronous I/O	
Dynamic invocation interface		Other:	

6. Roughly specify and prioritize the performance drivers in your ORB-based architecture.

Priority

- Average operation latency_____ ___
- Predictable operation latency_____ ___
- ORB operation rates (invocations/sec)_____ ___
- Typical message size _____ ___
- Total volume of data transferred_____ ___
- Complexity of interface data types_____ ___
- Other_____ ___
- Other_____ ___

7. Which of the following attributes influenced your selection of this particular ORB?

- Small memory footprint____
- Configurability (please identify specific configuration options that are important in your use of this ORB)____
- Reliability____
- Fault tolerance____
- Availability of specific CORBA services (please identify)
- Availability for specific hardware/OS platforms (please identify)
- Capability to support specific language bindings (identify the constraints)
- Availability of vendor support for specific language compilers and/or development environments (identify the constraints)
- Other?

8. What strengths and/or weaknesses of the ORB you are using have driven your decisions on selection of services for use?

INTER-ORB INTEROPERABILITY PROTOCOL (IIOP)

9. Have you used IIOP to communicate between this ORB and other ORBs, either real-time or non-real-time? Yes ____ No ____ (If you have not used IIOP, please skip to question 12.)

10. Please list the platforms/ORBs used for inter-ORB communications.

Use of IIOP				
ORB / Release #[15]	CPU type	OS (by release #)	Compiler (language & vendor)	No. of Units running this configuration

11. TCP/IP does not provide the predictability that is essential to hard real-time communications.

(a) Does the ORB you are using provide the capability to replace TCP/IP with another transport protocol for basic ORB operations? Yes___ No ___ (If "no", please skip to question 12.)

(b) If you have used the capability, please answer the following:

(1) With what protocol(s) have you replaced TCP/IP?

(2) Please identify the product and vendor that you used to replace TCP/IP.

(3) On a scale of 1 (very easy) to 5 (very difficult), please rate the ease with which you were able to accomplish this replacement.

(4) What impact(s) on ORB behavior and performance did you observe after the replacement of the protocol?

(5) Please comment on your experience regarding your use of this feature.

PRODUCT INSTALLATION AND USE:

12. Please choose the answer that most accurately describes your opinion about the ORB products.

	Strongly Agree	Agree	Disagree	Strongly Disagree
This product is easy to install and integrates easily into our development environment.				
This product integrates easily into our configuration management environment.				
I am satisfied with the general capability of this product.				
I am satisfied with the real-time features and behavior of this product.				
My organization has found it easy to design and implement software using this ORB.				
Compared with other ORBs, my organization has found this product to be easier to use than other products. (leave blank if not applicable)				

13. Please comment on aspects of product use that prompted the responses in the previous table.

14. Please identify any unexpected or surprising dependencies of the product on elements of the development or runtime environment(s).

PRODUCT SUPPORT:

15. Please indicate the number of problem reports you have written against the product in each of the following categories:

	Priority 1. Prevents accomplish-ment of mission capability	Priority 2. Degrades mission capability - no work-around	Priority 3 Degrades mission capability - workaround available	Priority 4 Annoyance but does not degrade an essential capability	Priority 5 All other errors
Last three months					
Last year					
Over life of product use					

16. Please choose the answers that most accurately describe your opinions about ORB vendor product support.

	Strongly Agree	Agree	Disagree	Strongly Disagree
I am satisfied with the technical content of a typical vendor response to a problem report				
I am satisfied with vendor responsiveness to critical problems (priorities 1 & 2).				
I am satisfied with vendor responsiveness to routine problems (priorities 3-5).				
The cost of product support is reasonable.				

What mechanisms do you use to report software problems to the vendor?

Email____ Telephone____ Fax____Overnight Mail_____ Regular Mail____

Other _____

Have you purchased any engineering services from the ORB vendor?

Yes ___ No ___ If "no", skip to question 22.

Please indicate the type of services purchased:

Consulting on design or implementation using the ORB___

Consulting on performance enhancement ____

Addition of specific services or features ____

Product ports to additional CPU, OS, or compiler____

Other _____

20. Please choose the answers that most accurately describe your opinions about ORB vendor engineering services.

	Strongly Agree	Agree	Disagree	Strongly Disagree
I am satisfied with the lead time required to purchase engineering services				
I am satisfied with the timeliness of delivery of the purchased services				
I am satisfied with the quality of the purchased engineering services				
The cost of engineering services is reasonable.				

21. Please elaborate on aspects of ORB product support or purchased engineering services with which you were particularly pleased or displeased.

22. Please rank order the usefulness of the following sources of information in resolving problems in design and / or implementation.

Source	Rank
Vendor online documentation	
Vendor printed reference manuals	
Vendor reference manuals on CD-ROM	
Vendor telephone support	
Newsgroups or mail subscription lists for users	
Online (web-accessible) logs of known problems and FAQs	
Vendor email support	
Third party manuals or web sites	
Source code	
Vendor-supplied code examples	
Other_____	

23. For each of the following, please choose the answers that most accurately describe your opinion about ORB vendor documentation.

	Strongly Agree	Agree	Disagree	Strongly Disagree
The documentation is technically accurate.				
The documentation includes a complete source of reference information.				
The documentation is clear and unambiguous.				
It is easy to find relevant topics.				
The documentation includes excellent instructional material.				
The cost of documentation is reasonable.				

24. Have staff members attended a vendor-supplied training course? If yes, please answer the following questions for each relevant course:

(c) Title of course:

(d) Length in days or hours:

(e) What topics did the training cover?

(f) For what kind of audience was the training suitable?

Developers_____ System Architects _____ Managers _____

25. For each of the following, please choose the answer that most accurately describes your opinion about ORB vendor training.

	Strongly Agree	Agree	Disagree	Strongly Disagree
The training was valuable.				
The course was well organized.				
The technical content was excellent.				
The course materials were clear and understandable.				
The instructor has knowledgeable.				
I recommend the training course to other developers implementing a system that use this ORB product.				
The cost of training is reasonable.				

26. Please add any comments you wish to make identifying strengths and weaknesses of the vendor-supplied training.

Active Page Record

Page Numbers	Revision Level	Revision Type (Added, Deleted)
i		
ii		
iii		
iv		
v		
vi		
vii		
viii		
1-1		
1-2		
2-1		
2-2		
3-1		
3-2		
3-3		
3-4		
3-5		
3-6		
3-7		
3-8		
3-9		
3-10		
3-11		
3-12		
3-13		
3-14		
3-15		
3-16		
3-17		
3-18		
3-19		

Page Numbers	Revision Level	Revision Type (Added, Deleted)
3-20		
3-21		
3-22		
3-23		
3-24		
3-25		
3-26		
3-27		
3-28		
3-29		
3-30		
3-31		
3-32		
3-33		
4-1		
4-2		
4-3		
4-4		
4-5		
4-6		
4-7		
4-8		
4-9		
5-1		
5-2		
5-3		
5-4		
5-5		
5-6		
5-7		
5-8		
6-1		

Page Numbers	Revision Level	Revision Type (Added, Deleted)
Glossary-1		
References-1		
Acknowledgements–1		
A-1		
A-2		
A-3		
A-4		
A-5		
A-6		
A-7		
A-8		
A-9		
A-10		
A-11		
A-12		
A-13		
A-14		
B-1		
B-2		
B-3		
B-4		
B-5		
B-6		
B-7		
APR-1		
APR-2		

Page Numbers	Revision Level	Revision Type (Added, Deleted)