IST-2001-37652

Hard Real-time CORBA

Title

Project Management Manual

For the HRTC Project

Authors

Ricardo Sanz (UPM)
Karl-Erik Årzén (LTH)
Miguel Segarra (SCI)
Thomas Losert (TUV)

Reference IST37652/004
Date 2002-12-12
Release 1.2
Status Final
Clearance Project

Partners

Universidad Politécnica de Madrid
Lunds Tekniska Högskola
Technische Universität Wien
SCILabs Ingenieros

Summary Sheet

IST Project 2001-37652
HRTC
Hard Real-time CORBA

Project Management Manual

For the HRTC Project

Abstract:

The present document describes all the mandatory and suggested management policies for the HRTC project.

HRTC Partners:

Universidad Politécnica de Madrid
Lunds Tekniska Högskola
Technische Universität Wien
SCILabs Ingenieros.

Release Sheet (1)

Release:	**0.1 Draft**
Date:	2002/09/09
Scope	Initial version
Sheets	All

Release:	**0.2 Draft**
Date:	2002/09/09
Scope	Additional Content
Sheets	All

Release:	**0.3 Draft**
Date:	2002/09/29
Scope	Modified planning and deliverable dates
Sheets	All

Release:	**0.4 Draft**
Date:	2002/11/12
Scope	Modified planning and deliverable dates
Sheets	All

Release:	**1.0 Final**
Date:	2002/11/12
Scope	Minor corrections
Sheets	All

Release:	**1.1 Final**
Date:	2002/11/30
Scope	Added issues on quality Assurance
Sheets	All

Release:	**1.2 Final**
Date:	2002/12/12
Scope	Modified structures and minor content added.
Sheets	29-45

Table of Contents

1 Introduction ***6***
1.1 Content 6
1.2 Basic Policy 6
1.3 Partner Communication 6
1.4 Definitions, acronyms and abbreviations 6

2 Project organization ***7***
2.1 Introduction 7
2.2 Management Roles 7
2.3 Technical Roles 8

3 Work structure ***9***
3.1 Project Workpackages 9
3.2 CORBA Control Systems (WP1) 10
3.2.1 Description 10
3.2.2 Tasks and Deliverables 11
3.3 Real-time Protocols (WP2) 11
3.3.1 Description 11
3.3.2 Tasks and Deliverables 12
3.4 Robot Control Testbed (WP3) 13
3.4.1 Description 13
3.4.2 Tasks and Deliverables 13
3.5 Process Control Testbed (WP4) 14
3.5.1 Description 14
3.5.2 Tasks and Deliverables 14
3.6 Dissemination and Standardization (WP5) 15
3.6.1 Description 15
3.6.2 Tasks and Deliverables 15

4 Project Plan ***17***
4.1 Project Planning 17
4.2 Complete list of deliverables 17

5 Project Meetings ***19***
5.1 Meeting Types 19
5.2 Meeting Policies 19
5.3 Meeting Schedule 20

6 Basic Project Policies ***21***
6.1 Making Decissions 21
6.2 Mail interchange 21
6.3 Documentation Standards 21
6.4 Quality Assurance 22
6.5 Reporting 23

7 Online resources ***24***
7.1 Project Website 24
7.2 Project Mailing Lists 25

8 Documentation ***26***
8.1 Introduction 26
8.2 General procedures for documentation 26
8.3 Documentation identification 28
8.4 Document Format 28

9 Tools ***29***
9.1 Documentation 29
9.2 Specification and design 29
9.3 Coding 29
9.4 Configuration management and version control 29

10 Document Templates ***30***
10.1 Project Long Report 30
10.2 Project Short Report 30
10.3 Slides Dark 30
10.4 Slides Light 30
10.5 Template availability 30

11 Glossary ***31***

1 Introduction

1.1 Content

This document is the Project Management Plan that describes common guidelines for project development, establishing mechanisms and tools for information exchange among the partners and templates for internal and deliverable documents.

1.2 Basic Policy

It is fundamental to agree upon a relatively large set of procedures and tools so as to minimize possible but undesirable variability among the several partners. On the other hand excessive regulations can cause unnecessary limitations for future developments. With that in mind, some procedures were given more detailed coverage than others.

1.3 Partner Communication

Communication among partners is treated taking into account the physical distance among the different partners, which obliges the use of e-mail, fax, telephone and mail.

1.4 Definitions, acronyms and abbreviations

This document contains a detailed glossary where the terms employed in the project are well defined (See Section 10).

2 Project organization

2.1 Introduction

HRTC partners have wide experience in international joint projects in the framework of EU funded research and development. HRTC is an RTD project but with a strong research orientation. This is the main reason for the technical structure, the composition of the consortium and the management organization.

2.2 Management Roles

UPM performs the overall co-ordination, control and supervision of the HRTC Project. As coordinator, UPM is responsible for the communication with the EC, for the compilation of project management reports, and for the transfer of payments to the contractors.

For this reason, the **General Project Manager**, who is in charge of the overall control of the project progress, and all official communication with the EC, will be a person from UPM, namely Ricardo Sanz (Ricardo.Sanz@etsii.upm.es).

Each partner has nominated its own **Project Manager**, who is responsible for the performance of that partner's **Project Team**. This person will control the participation of that partner's Project Team in the HRTC Project activities, and will co-ordinate that participation with the rest of the partners.

Organization	*Project Manager*	*E-mail*
Technical University of Madrid	Ricardo Sanz	Ricardo.Sanz@etsii.upm.es
Lund University	Karl-Erik Årzën	karlerik@control.lth.se
Vienna Technical University	Hermann Kopetz	hk@vmars.tuwien.ac.at
SCILabs Ingenieros	Miguel Segarra	mjsegarra@scilabs.es

The General Project Manager and each individual partner Project Manager constitute the **Management Committee** that takes all decisions regarding the overall project coordination and control.

The e-mail address to be used for the HRTC management committee is <mc@hardrealtimecorba.org>.

The Project Manager from SCILabs will continuously guarantee that the necessary alignment between the project activities performed by the Project Teams, and the plans set by that partner for the future industrial exploitation of the project results, is being maintained.

2.3 Technical Roles

Each workpackage is lead by one organization and has a responsible from that organization called the WorkPackage Manager.

Workpackage	*Workpackage Manager*	*E-mail*
WP1: CORBA Control Systems	Karl-Erik Arzen	karlerik@control.lth.se
WP2: Real-time Protocols	Thomas Losert	thomas@vmars.tuwien.ac.at
WP3: Robot Control Testbed	Klas Nilsson	klas@cs.lth.se
WP4: Process Control Testbed	Manuel Rodríguez	mrod@diquima.upm.es
WP5: Dissemination	Miguel Segarra	mjsegarra@scilabs.es
WP6: Management	Ricardo Sanz	Ricardo.Sanz@etsii.upm.es

There is also an e-mail distribution list for the workpackage managers: wpm@hardrealtimecorba.org.

3 Work structure

3.1 Project Workpackages

The work is organized into six workpackages, which in turn are composed by tasks. For each workpackage/task, one partner is nominated as workpackage/task leader. The leader will be in charge of controlling the work in the workpackage/task in a way that ensures an in-time availability of the envisaged workpackage/task results.

Workpackage	Task	UPM	Lund	Wien	SCI	P·M
WP1 CORBA Control Systems		12,5	14	3	1,5	**31**
	T1.1 CCS Domain Analysis	4	8	1	0,5	13,5
	T1.2 CCS Domain Architectures	4	4	1	0,5	9,5
	T1.3 CCS Engineering Process	4,5	2	1	0,5	8
WP2 HRT Protocols		2	2	4	5	**13**
	T2.1 Protocols for Real-time Control		1	2	2	5
	T2.2 HRT Pluggable CORBA Protocol	2	1	2	3	8
WP3: Robot Control Testbed		5	10	1	4	**20**
	T3.1 RCT Requirements Specification	1	1		1	3
	T3.2 RCT Design	1	1			2
	T3.3 RCT Procurement	1	1			2
	T3.4 Non HRTP RCT implementation	1	3		1	5
	T3.5 HRTP RCT Implementation	1	2	1	2	6
	T3.6 RCT Testing		1			1
	T3.7 RCT Documentation & Eval		1			1
WP4: Process Control Testbed		12	5	1	4	**22**
	T4.1 PCT Requirements Specification	1	1		1	3
	T4.2 PCT Design	1	1			2
	T4.3 PCT Procurement	1	1			2
	T4.4 Non HRTP PCT implementation	5	1		1	7
	T4.5 HRTP PCT Implementation	2	1	1	2	6
	T4.6 PCT Testing	1				1
	T4.7 PCT Documentation & Eval	1				1
WP5 Dissemination and Exploitation		7,5	4	2,5	3,5	**17,5**
	T5.1 Dissemination Planning	0,5				0,5
	T5.2 OMG Standardization	2,5	1	1,5	1	6
	T5.3 Publications	2	2,5	0,5	0,5	5,5
	T5.4 HRTC Web site	1	0,5	0,5	0,5	2,5
	T5.5 Exploitation and Use Planning	1,5			1,5	3
WP6 Management		7	2	2	1	**12**
	T6.1 Project Coordination	3				3
	T6.2 Project Management	3	1,75	1,75	0,75	7,25
	T6.3 Project Assessment	1	0,25	0,25	0,25	1,75
	M·M Totals	**46**	**37**	**13,5**	**19**	**115,5**

Task Leader
WP Coordinator

Total Effort **115,5**

Efforts are in Person·Months

The following sections are extracted from the *HRTC Annex 1 – Description of Work*.

3.2 CORBA Control Systems (WP1)

3.2.1 Description

The theoretical work in the project is concerned with the understanding and modeling of distributed real-time systems focusing on control applications based on CORBA technology. The objective of this work is the scientific analysis of the problem of closing control loops using distributed object technology in the framework of CORBA based systems (CORBA Control Systems: CCS).

As it is well known, a hardware/software platform for good controllers needs to be end-to-end predictable for engineers to guarantee controller behavior. This means that system predictability can only be obtained if all the elementary components do have this property. In the case of CCS this means computer hardware, network hardware, operating systems, network software, middleware and application proper.

It is out of the scope of the project to provide an implementation of all the elements involved but we will try to identify the core issues at the level of component interfaces.

Control architectures will be explored as a mechanism for easy design and construction of complex controllers. The work will address the CCS domain defining potential object-based architectures for control systems implementation and specifying theoretical tools to support the design and validation of these systems.

Testbeds developed in WP3 and WP4 will be modeled and analyzed to identify real-time behavior departure points.

3.2.2 Tasks and Deliverables

T1.1 CCS Domain Analysis. Domain requirement analysis of the field of CORBA-based control systems. This task will identify requirements for CCS in a broad spectrum (*i.e.* a domain).

T1.2 CCS Domain Architectures. Specification of a core set of architectures for CCS addressing a wide collection of potential applications of the technology.

T1.3 CCS Engineering Process. Elaboration of an engineering handbook for the construction of CCS.

D1.1 CCS Domain Analysis (D/ULund)
D1.2 CCS Domain Architectures (D/UPM)
D1.3 CCS Engineering Handbook (D/UPM)

3.3 Real-time Protocols (WP2)

3.3.1 Description

The base CORBA interoperability protocol is called GIOP (General Inter-ORB protocol). It is a protocol for ORB interaction and the main realization of it is called IIOP (Internet Inter-ORB Protocol). IIOP, running atop TCP/IP, is not adequate for every deployment condition and application requirement (in particular it is unsuitable for hard real-time applications). To be more adaptable to special conditions the ORB can replace the transport if necessary. Using the pluggable transports interface the broker can bring into play different transports for different purposes.

The first work to do in this workpackage is the study of the problem of hard real-time communication in control systems (industrial networking, fieldbuses, time-triggered protocols, etc.) and an analysis of it in the framework of CORBA (IIOP & extensible transport framework).

The next work to be done in this workpackage is the selection of adequate networking hardware and software and the construction of a real-time protocol to be used in the transport layer of a CORBA Broker. Alternatives will be explored if possible (for example building GIOP over TTP and GIOP over RT-Ethernet). This real-time protocol will be built to be integrated with the object request broker using a pluggable transports interface.

The real-time protocol will be used in the testbeds and evaluated.

3.3.2 Tasks and Deliverables

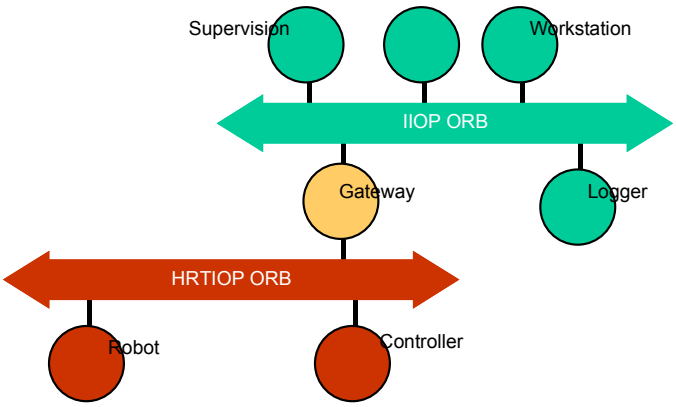


Figure 1: Overview of the envisioned final Robot Control Testbed. The final structure will be detailed in deliverable *D3.2 RCT Design*. ORBs using real-time transports (for example over RT-Ethernet or TTP) and ORBs using a conventional IIOP transport can be gatewayed to provide an heterogeneous transports testbed.

T2.1 Protocols for Real-time Control. Analysis of the domain of hard real-time communication in control systems.

T2.2 HRT Pluggable CORBA Protocol. Design and implementation of a real-time interoperability protocol for CORBA ORBs.

D2.1 Analysis of Protocols for Real-time Control (D/TUWien)
D2.2 HRT Protocol Specification (D/SCILabs)
D2.3 HRT Protocol (S/SCILabs)

3.4 Robot Control Testbed (WP3)

3.4.1 Description

The Robot Control Testbed (RCT) is an experimental platform used to identify requirements and perform tests of CCSs.

The RCT will be built in the Automatic Control Laboratory of Lund University and will be composed of a robot, embedded control computers, specialized networks (defined in WP2), conventional networks and conventional workstations.

The rationale of the RCT is the interest of the consortium in addressing controllers with *tight timing* requirements and evaluation RT-CORBA suitability and in particular the advantages of using real-time protocols. Robot control serves also as a good demonstration platform due to its visual appeal.

3.4.2 Tasks and Deliverables

T3.1 RCT Requirements specification. Specification of requirements for RCT based on the domain partitioning done in T1.1.

T3.2 RCT Design. Design of the RCT based on available equipment and requirements expressed in D3.1 (installation and control systems)

T3.3 RCT Procurement. Procurement of equipment for RCT, including production of a related description.

T3.4 Non HRTP RCT Implementation. Implementation of control algorithms using RT CORBA

T3.5 HRTP RCT implementation. Implementation of control algorithms using RT CORBA and the new transport

T3.6 RCT Testing
T3.7 RCT Documentation & Evaluation
D3.1 RCT Requirements specification (D/ULund)

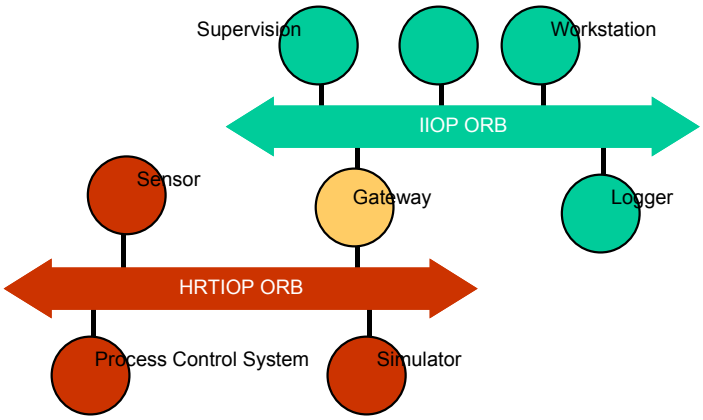


Figure 2: Final envisioned structure of the Process Control Testbed. The final structure will be detailed in deliverable D4.2 PCT Design.

D3.2 RCT Design (D/ULund)
D3.3 RCT Procurement (D/ULund)
D3.4 Non HRTP RCT Implementation (T/ULund)
D3.5 HRTP RCT implementation (T/ULund)
D3.6 RCT Testing (D/ULund)
D3.7 RCT Documentation (D/ULund)

3.5 Process Control Testbed (WP4)

3.5.1 Description

The Process Control Testbed (PCT) is an experimental platform used to identify requirements and perform tests of CCSs.

The PCT will be built in the Automatic Control Laboratory of Polytechnic University of Madrid and will be composed of a simulated process, an embedded sensor, a process control system, specialized networks (defined in WP2), conventional networks and conventional workstations.

The rationale behind the PCT is the interest of the consortium in addressing controllers with *complexity+legacy+heterogeneity* requirements.

3.5.2 Tasks and Deliverables

T4.1 PCT Requirements specification. Specification of requirements for PCT based on the domain partitioning of D1.1.

T4.2 PCT Design. Design of the PCT based on available equipment and requirements expressed in D4.1 (installation and control systems).

T4.3 PCT Procurement. Procurement of equipment for PCT.

T4.4 Non HRTP PCT Implementation. Implementation of control algorithms using RT CORBA.

T4.5 HRTP PCT implementation. Implementation of control algorithms using RT CORBA and the new transport.

T4.6 PCT Testing.

T4.7 PCT Documentation & Evaluation

D4.1 PCT Requirements specification (D/UPM)
D4.2 PCT Design (D/UPM)
D4.3 PCT Procurement (D/UPM)
D4.4 Non HRTP PCT Implementation (T/UPM)
D4.5 HRTP PCT implementation (T/UPM)
D4.6 PCT Testing (D/UPM)
D4.7 PCT Documentation (D/UPM)

3.6 Dissemination and Standardization (WP5)

3.6.1 Description

Workpackage 5 is focused on dissemination and exploitation activities. This concrete project, being more related with standardization than with concrete technological developments, will put more emphasis in the dissemination side of the work. In particular, the main activity will be related with fostering, inside the OMG, the generation of specifications for hard real-time systems based on CORBA technology.

The OMG's Technology Adoption Process[1] is a well established specification development process.

The work planned is the collaboration with the OMG in the creation of a task force (TF) on Hard Real-time Systems, the elaboration of an RFI, the elaboration of an RFP and the elaboration of a concrete proposal. The actual work that will be done inside the project will depend on OMG timing issues that the HRTC consortium cannot control.

Most of this activity will be done by means of participation in OMG technical meetings and workshops.

[1] http://www.omg.org/gettingstarted/processintro.htm

3.6.2 Tasks and Deliverables

T5.1 Dissemination Planning. Elaboration of deliverable *D5.1 Dissemination Plan*

T5.2 OMG Standardization. Collaboration with OMG TCs, elaboration of documents, attendance to OMG Technical Meetings and meeting reporting.

T5.3 Publications. Elaboration of journal, symposia and academic publications.

T5.4 HRTC Web site. Construction of an HRTC web site.

T5.5 Exploitation and Use Planning. Elaboration of deliverable *D5.6 Exploitation and Use Plan.*

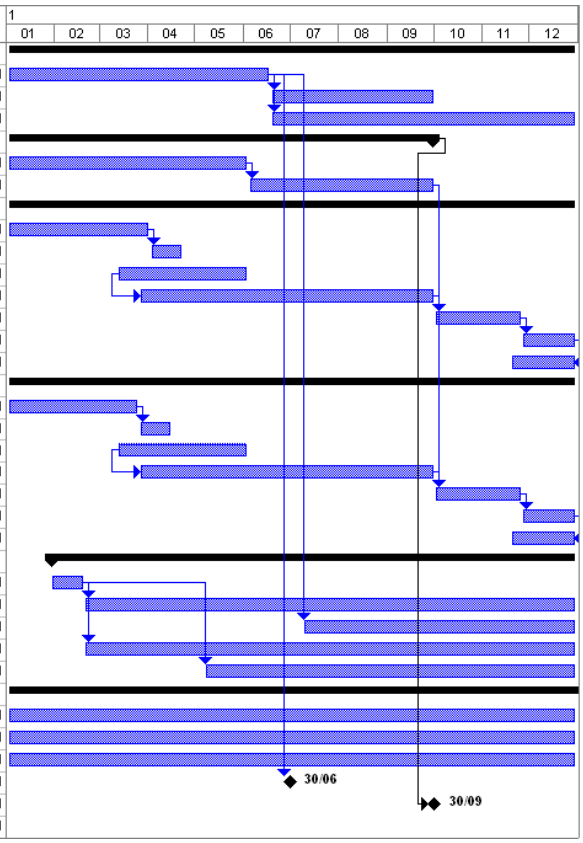
D5.1 Dissemination Plan (D/UPM)
D5.2.x OMG Documents (D/TUWien)
D5.3.x Publications (D/Px)
D5.4 Project Web Page (S/SCILabs)
D5.5 Draft Exploitation and Use Plan (D/SCILabs)
D5.6 Exploitation and Use Plan (D/SCILabs)

4 Project Plan

4.1 Project Planning

The project has suffered an initial delay due to the starting date coincident with holidays. This is the revised Gantt diagram of the project:

Id	Nombre de tarea	Duración
1	**1 CORBA Control Systems**	**260 d**
2	1.1 CCS Domain Analysis	120 d
3	1.2 CCS Domain Architectures	75 d
4	1.3 CCS Engineering Process	140 d
5	**2 HRT Protocols**	**195 d**
6	2.1 Protocols for Real-time Control	110 d
7	2.2 HRT Pluggable CORBA Protocols	85 d
8	**3 Test 1: Robot Control**	**260 d**
9	3.1 RCT Requirements specification	65 d
10	3.2 RCT Design	15 d
11	3.3 RCT Procurement	60 d
12	3.4 Non HRTP RCT Implementation	135 d
13	3.5 HRTP RCT implementation	40 d
14	3.6 RCT Testing	25 d
15	3.7 RCT Documentation	30 d
16	**4 Test 2: Process Control**	**260 d**
17	4.1 PCT Requirements specification	60 d
18	4.2 PCT Design	15 d
19	4.3 PCT Procurement	60 d
20	4.4 Non HRTP PCT Implementation	135 d
21	4.5 HRTP PCT implementation	40 d
22	4.6 PCT Testing	25 d
23	4.7 PCT Documentation	30 d
24	**5 Dissemination**	**240 d**
25	5.1 Dissemination Planning	15 d
26	5.2 OMG Standardization	225 d
27	5.3 Publications	125 d
28	5.4 HRTC Web site	225 d
29	5.5 Exploitation and Use Planning	170 d
30	**6 Management**	**260 d**
31	6.1 Project Coordination and Reporting	260 d
32	6.2 Project Management	260 d
33	6.3 Project Assessment	260 d
34	6.4 Milestone 1: Domain Analysis	0 d
35	6.5 Milestone 2: HRT Protocols	0 d
36	6.6 Milestone 3: Final Results	0 d



4.2 Complete list of deliverables

The following list contains the document deliverables as foreseen in the Annex 1 to the HRTC contract – "Description of Work" with the modification in delivery date due to initial project delay.

Id	Deliverable Name	Lead	Effort	Type	Sec	Delivery
Ordered by delivery date						
D5.1	*Dissemination Plan*	UPM	0,5	D	*Res*	M3
D6.1	*Project Management Manual*	UPM	0.5	D	*Res*	M3
D3.1	*RCT Requirements specification*	ULUND	3	D	*Pub*	M4
D4.1	*PCT Requirements specification*	UPM	3	D	*Pub*	M4
D6.2	*Evaluation Plan*	UPM	0.5	D	*Res*	M4
D2.1	*Protocols for Real-time Control*	TUWien	5	D	*Pub*	M5
D3.2	*RCT Design*	ULUND	2	D	*Res*	M5
D4.2	*PCT Design*	UPM	2	D	*Res*	M5
D6.3	*Quarterly Report M3*	UPM	0.5	D	*Res*	M5
D3.3	*RCT Procurement*	ULUND	2	D	*Res*	M6
D4.3	*PCT Procurement*	UPM	2	D	*Res*	M6
D1.1	*CCS Domain Analysis*	ULUND	12.5	D	*Pub*	M6
D2.2	*HRT Protocol Specification*	SCILabs	2	D	*Int*	M6
D5.5	*Draft Exploitation and Use Plan*	SCILabs	1	D	*Int*	M6
D6.4	*Periodic Report M6*	UPM	1	D	*Res*	M6
D1.2	*CCS Domain Architectures*	UPM	9.5	D	*Pub*	M9
D2.3	*HRT Protocol*	SCILabs	6	S	*Int*	M9
D3.4	*Non HRTP RCT Implementation*	ULUND	6	T	*Int*	M9
D4.4	*Non HRTP PCT Implementation*	UPM	6	T	*Int*	M9
D6.5	*Quarterly Report M9*	UPM	0.5	D	*Res*	M9
D3.5	*HRTP RCT implementation*	ULUND	6	T	*Int*	M11
D4.5	*HRTP PCT implementation*	UPM	6	T	*Int*	M11
D1.3	*CCS Engineering Handbook*	UPM	8	D	*Res*	M12
D3.6	*RCT Testing*	ULUND	1	D	*Res*	M12
D3.7	*RCT Documentation*	ULUND	1	D	*Res*	M12
D4.6	*PCT Testing*	UPM	1	D	*Res*	M12
D3.7	*PCT Documentation*	UPM	1	D	*Res*	M12
D5.4	*HRTC Project Web Page*	SCILabs	2.5	D	*Pub*	M12
D5.6	*Exploitation and Use Plan*	SCILabs	1.5	D	*Int*	M12
D6.6	*Periodic Report M12*	UPM	0.5	D	*Res*	M12
D6.7	*Project Evaluation Report*	UPM	1	D	*Res*	M12
D6.8	*Final Report*	UPM	2	D	*Res*	M12
Out of order						
D5.2.x	*OMG Documents*	TUWien	6	D	*Pub*	Mx
D5.3.x	*Publications*	Px	5.5	D	*X*[2]	Mx
Types	*D* *Document* *S* *Software = Software + Documentation* *T* *Testbed = Hardware+Software+Documentation*					

[2] The security/clearance level of these documents will be assigned by the General Project Manager in accordance with the HRTC Management Committee guidelines.

5 Project Meetings

5.1 Meeting Types

There will be two types of project meetings:

Plenary Meetings will deal with the definition of goals, and the control of the intermediate project results. Generally, there will be a Plenary Meeting before each Project Review, but also at other times, depending on the current project situation. Plenary Meetings will take place about every three months. All Project Managers (*i.e.* the Management Committee) will take part in these meetings. All those other persons considered necessary by the Management Committee will also take part (but not vote) in these meetings.

Technical Meetings will deal with the discussion, harmonization and clarification of scientific-technical decisions to be made. Apart from that, Technical Meetings will be the forum for presenting intermediate results. Technical meetings will take place every three months.

Of course, additional meetings will be organized during the project life to cope with unexpected problems if necessary.

5.2 Meeting Policies

When a technical meeting coincides with a plenary meeting, the technical one will be held in first place, and its results and decisions will be reflected in the short term planning and minutes of the plenary meeting. If it is not coincident with a plenary meeting, a technical short term planning will be generated after the end of the technical discussions.

This project has a strong commitment with OMG activities and other international events (like ISORC, WORDS and ERTDOS) and so, meetings will be scheduled taking into account these international gatherings.

In general, the project meetings will be held cyclically at the consortium partners' premises, in order to distribute the traveling expenses as equally as possible among all the partners.

5.3 Meeting Schedule

The meetings of the project will be as follows:

Date	Milestone	Duration	Place	Type
July 2002		½ Day	Barcelona	Management
September 2002		2 days	Vienna	Plenary
January 2003	M6	2 Days	Brussels	Plenary + Review
April 2003	M9	2 Days	Madrid	Technical
July 2003	M12	3 Days	Lund	Plenary + Review

The consortium is planning the realization of an open workshop in coincidence with the final meeting.

6 Basic Project Policies

6.1 Making Decissions

Decisions will be made on a voting base. In case of balanced votes, the partner responsible for the activity to be discussed (the workpackage/task leader) will decide. For each meeting, minutes will be written, circulated to the partners for comment, and formally approved by the consortium.

The General Project Manager will inform the EC Project Officer about all topics discussed and decided in the meetings that are to be communicated to the EC. If necessary, the General Project Manager can decide to deliver a copy of the corresponding meeting minutes to the EC.

An important topic of project meetings is the supervision of all activities in relation to the Workplan, regarding the time schedule as well as concerning the planned and consumed resources. In case that intermediate changes to the Workplan are necessary, they will be discussed in the Plenary Meetings; the result of these discussion will be formally reported to the EC for approval if they modify the content of the milestones.

6.2 Mail interchange

The proper way of interacting is by e-mail. There are three different mail lists of different scopes:

- All personnel involved in HRTC
- Workpackage Managers
- Management Committee

All e-mails should have a "Subject" starting with the project acronym: "HRTC" to facilitate identification and management of e-mail.

6.3 Documentation Standards

Project documents should use the document templates available at the project website.

6.4 Quality Assurance

Due to the exploratory nature of the work performed in the project it is not necessary to perform exhaustive quality assurance. There will be a simple procedure for document deliverable quality control.

All deliverables should be approved by all the members of the management committee. This means that besides the authors of the document, all the partners are responsible for quality assurance of other's deliverables.

Bu to avoid dilution of quality assurance responsibility each deliverable is assigned to a concrete partner to guarantee that quality checks are performed (the concrete person to perform the quality assurance control work will be assigned by each project manager).

The list of assignments is as follows:

Id	Deliverable Name	Lead	Quality Lead
D5.1	*Dissemination Plan*	UPM	SCI
D6.1	*Project Management Manual*	UPM	ULUND
D3.1	*RCT Requirements specification*	ULUND	UPM
D4.1	*PCT Requirements specification*	UPM	ULUND
D6.2	*Evaluation Plan*	UPM	SCI
D2.1	*Protocols for Real-time Control*	TUWien	ULUND
D3.2	*RCT Design*	ULUND	UPM
D4.2	*PCT Design*	UPM	ULUND
D6.3	*Quarterly Report M3*	UPM	TUWien
D3.3	*RCT Procurement*	ULUND	TUWien
D4.3	*PCT Procurement*	UPM	TUWien
D1.1	*CCS Domain Analysis*	ULUND	UPM
D2.2	*HRT Protocol Specification*	SCILabs	ULUND
D5.5	*Draft Exploitation and Use Plan*	SCILabs	UPM
D6.4	*Periodic Report M6*	UPM	SCI
D1.2	*CCS Domain Architectures*	UPM	UPM
D2.3	*HRT Protocol*	SCILabs	TUWien
D3.4	*Non HRTP RCT Implementation*	ULUND	UPM
D4.4	*Non HRTP PCT Implementation*	UPM	ULUND
D6.5	*Quarterly Report M9*	UPM	TUWien
D3.5	*HRTP RCT implementation*	ULUND	UPM
D4.5	*HRTP PCT implementation*	UPM	ULUND
D1.3	*CCS Engineering Handbook*	UPM	SCI
D3.6	*RCT Testing*	ULUND	UPM
D3.7	*RCT Documentation*	ULUND	UPM
D4.6	*PCT Testing*	UPM	ULUND
D3.7	*PCT Documentation*	UPM	ULUND
D5.4	*HRTC Project Web Page*	SCILabs	UPM
D5.6	*Exploitation and Use Plan*	SCILabs	UPM
D6.6	*Periodic Report M12*	UPM	SCI
D6.7	*Project Evaluation Report*	UPM	TUWien
D6.8	*Final Report*	UPM	SCI
Out of order			
D5.2.x	*OMG Documents*	TUWien	UPM
D5.3.x	*Publications*	Px	Py

6.5 Reporting

Periodic reports of the project will be done by the general project manager using contributions from all the partners. As with any other deliverable, they must be approved by the Management Committee before external release.

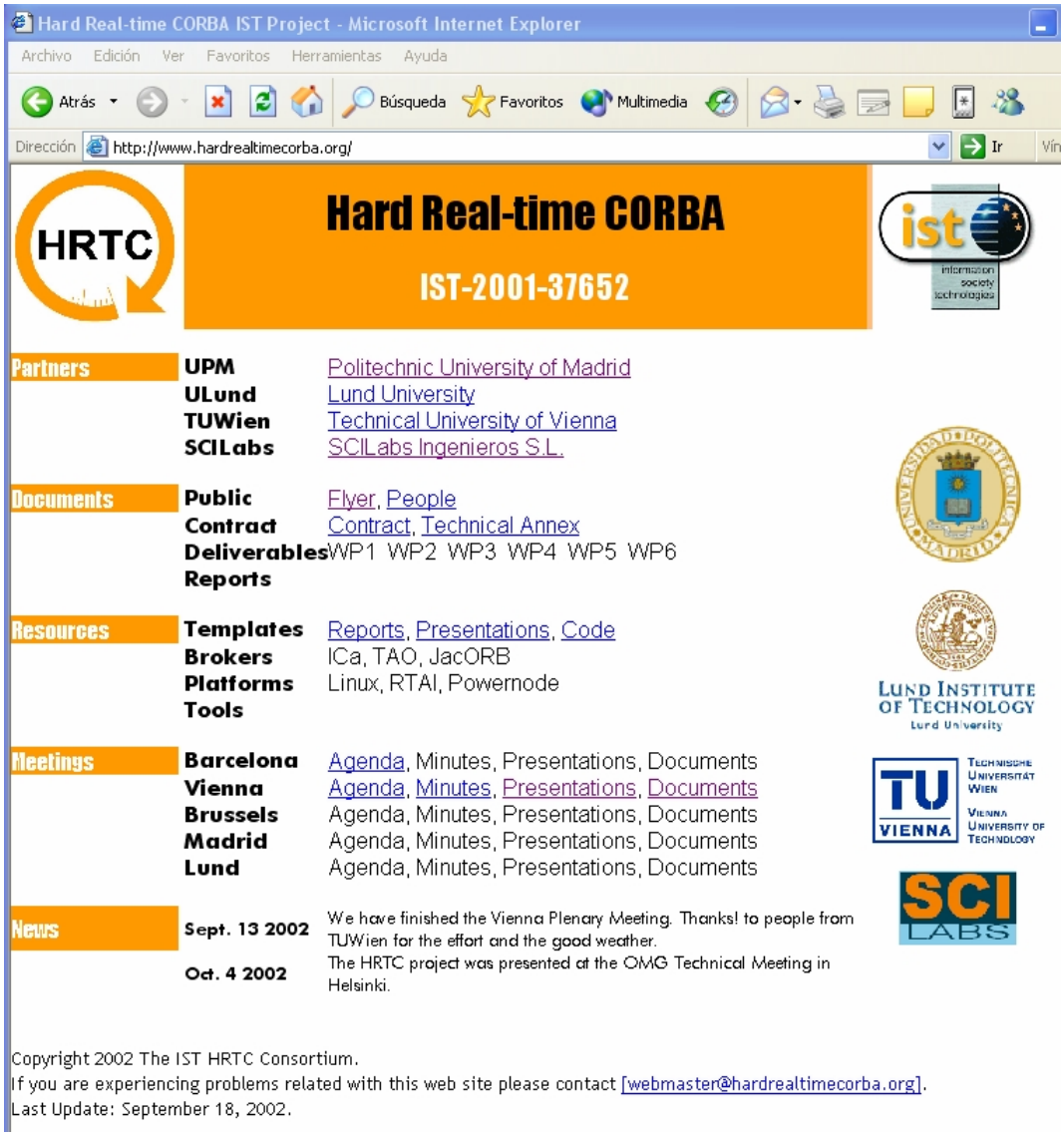
7 Online resources

7.1 Project Website

This website will contain public information as well as private documentation, both internal to the Consortium and deliverable documents. The public part will be used for dissemination purposes during the life of the project, including the later exploitation phase. The private part will serve as a repository of the internal documentation to be used by the Consortium members. Access to this part will be restricted.

www.HardRealTimeCORBA.org

The username/password needed for the private part will be controlled by the website manager.



7.2 Project Mailing Lists

All the people involved in the project:

all@hardrealtimecorba.org

The project Management Committee:

mc@hardrealtimecorba.org

The Project WorkPackage Managers:

wpm@hardrealtimecorba.org

8 Documentation

8.1 Introduction

Many products (documents and code) will be generated during the project. They will change even after initial approval. That should be seen as something natural and good as it means that the vision is improving.

The distributed nature of the development makes tracking of these changes particularly important. For that, a central repository will be maintained by UPM containing all the baseline documents and code as well as all outdated versions. The baselines are documents already reviewed and corrected accordingly, as well as code already tested.

8.2 General procedures for documentation

All documents generated will be compliant with the general document style and format defined in templates. All documents maintain a history of revisions containing their respective version number using the specified scheme. The current version number appears in the document title page and page headings.

For each of the project documents, one partner has been nominated as the responsible for its preparation and maintenance, while the other partners participating in the workpackage are committed to properly contribute to the preparation, modifications, approval and revision of the document, in accordance with the document Annex 1, "Description of Work".

The general procedures to be followed for the preparation, approval, revision and modifications of the documents are:

- The responsible partner will drive the preparation of the document, defining its basic structure and the contributions which are necessary from the other partners.

- The responsible partner will request, obtain and integrate the contributions from the participating partners, thus producing a first preliminary version of the document that will be distributed to all the participating partners.

- Each participating partner will revise this preliminary version of the document, and must distribute to the other partners the outcomes of this revision. These could be a set of comments, requests for modifications, some new contributions to be included, and even their approval conditioned to the fact that some changes proposed by him are included in the document.

- The responsible partner will co-ordinate this revision and the corresponding discussions around the document, as well as the modifications to be performed as a consequence of the comments provided. He will produce a new preliminary version any time he considers it to be necessary or convenient to efficiently complete the process towards the required approval of the document by all the participating partners.

- The participating partners will revise any new preliminary version of the document issued by the responsible partner. They must provide additional comments whenever necessary, and must make explicit their approval to the document as soon as a distributed preliminary version is acceptable for them.

- When a preliminary version of the document has been approved by all the participating partners, the responsible one will ask the management committee to change the status of the document from preliminary to final. Then the preparation of the document is considered to be finished, and its maintenance starts.

- Any partner who identifies the need for a modification in an approved document, must notify this need to the others, explaining and justifying that need and describing the suggested modification.

- Whenever a modification to an approved document is proposed, the responsible partner will co-ordinate its discussion among the participating partners, and will perform the necessary activities to obtain a new version of the document including the modification, formally approved by all the partners, if the modification is accepted. Whenever it is possible, the responsible partner will manage the proposals for modifications grouping them it the most convenient way, thus reducing the effort to be devoted for the necessary maintenance of the document.

- In the unusual case that, after revising and discussing a new version of a document, or a modification to a document approved in the past, the partners are not able to come to a full agreement for the approval, the corresponding decision on the approval or rejection will be taken by the partner responsible for the document being discussed.

- Any new release of the document must be approved by the Management Committee before being considered final.

8.3 Documentation identification

All project document cover sheets should contain the following information that will be gathered by the coordinator:

Title	Title of the document
Subtitle	Subtitle
Authors	Complete list of authors with affiliations
Reference	Document reference
Date	Date of release
Release	Release Number
Status	Status of the document
Clearance	Who can read the document

The document ***reference*** is a unique identification code, which enables further reference to the document at any later point in time. The format of the document reference will be IST37652/XXX, where XXX is a single number identifying the deliverable. The reference will be the same for all the releases of the document. If anyone wants to refer to a specific release he will use a ***complete reference*** that is composed by a reference and the release number IST37652/XXXvY.Z (for example this document is IST37652/004v1.2)

The general project manager will generate and keep updated the list of documents (including releases).

The status can be ***draft***, ***final*** (after approval of the MC) and ***approved*** (after approval by the commission review team).

The clearance can be ***consortium*** (distribution restricted to the HRTC Consortium) ***project*** (HRTC Consortium and review commission), ***restricted*** (list specified in the cover sheet) and ***public***.

8.4 Document Format

Documents should use the templates available at the project website.

Graphics required for a document will be included in the document itself, constituting a self-contained unit. Therefore, a single file should include both the text and the graphics.

9 Tools

In this section, all the tools used should be mentioned and use details should be given for the tools used by more than one partner. Each workpackage will be allowed to select the tools that best suit their needs.

This list of specifications will progress as necessary.

9.1 Documentation

For an efficient and error-free data exchange among the partners, common desktop software tools (word processor, spreadsheet, etc.) must to be used by all of them.

As most of the partners are using the same tools, the Microsoft Office 2000/XP tools (Word, Excel, PowerPoint, etc) have been selected for that purpose.

9.2 Specification and design

UML tools will be used to do graphic modeling. TBD.

9.3 Coding

The software tools will be identified at the workpackage level. TBD.

9.4 Configuration management and version control

CVS will be used to keep track of document and common code releases. A CVS repository has been set up at UPM.

10 Document Templates

10.1 Project Long Report

Used for deliverables and long reports.

10.2 Project Short Report

Used for minutes and short reports.

10.3 Slides Dark

PowerPoint template with dark background.

10.4 Slides Light

PowerPoint template with clear background.

10.5 Template availability

All these templates are available at the project website.

11 Glossary

A

ABI - Application Binary Interface, which defines how programs should interface with the operating system, including specifications such as executable format, calling conventions, and chip-specific requirements.

API - Application Programming Interface. An API is a set of functions that can be called to perform actions.

ATM - Asynchronous Transfer Mode. A packet switched network protocol using a pre-established connection route.

Attributes - Data that is stored in the object.

B

BSP - Board Support Package, typically referring to the low-level code or scripts that build programs running on a particular chip on a particular circuit board. Also refers to the ROM that boots an RTOS onto a specific board. Exact meaning varies.

Build - The process of configuring, compiling, and linking a set of tools. Also used as a noun, to denote the results of the process.

Bus - A physical connection between components of a single computer system. Examples include VME, PCI, etc.

C

Callbacks - Programming functions that are executed when a specific type of event occurs.

CCM - CORBA Component Model.

CSS – CORBA Control Systems.

Client - An object, component, or application that makes requests for services from other objects, components, or applications - implementation objects or servers. An object, component, or application can be a client for some requests and a server for other requests.

Client/Server model - A distributed application architecture where clients are the users of services provided by the servers.

COFF - Common Object File Format. This debug format appeared with Unix SVR3, formerly common for Unix, and still used by some embedded systems. The Microsoft PE format for Windows is based on COFF.

COFF debugging - The debug format that is defined as part of the COFF specification.

COM - Common Object Model

Compiler - A tool that translates high-level source code in a language such as C or Pascal into machine-executable programs. The term may also refer specifically to the tool that translates from source to assembly language.

CORBA - Common Object Request Broker Architecture is a standard for interoperability in heterogeneous computing environments. It enables applications to cross boundaries of different computing machines, operating systems, and programming languages. It specifies how client applications can invoke operations on server objects.

CORBA Object - A "virtual" entity capable of being located by an ORB and having client requests delivered to it. A CORBA object is identified, located, and addressed by its object reference. Within the context of a request invocation, the CORBA object to which the request is sent is called the "target object".

CORBAServices - CORBAServices are the specifications of the objects that provide basic assistance to developers - they augment the ORB. Some services include Naming, Event Notification, Life Cycle, Transaction, and Properties to name a few.
Criticality of response - A hard real-time system has an absolute response value, dictated by the environment, whereas a soft system specifies an average value, determined by the business or market.

Cross-compiler - A compiler that generates code for a different environment than which it is run.

Cross-development - Development of a program in one environment (host) which is designed to run in a different environment (target).

CVS - Concurrent Version System, a free source version control.

D

Data types - The description of the kinds of data stored, passed and used.

DCOM - Distributed Component Object Model; Networked OLE, a system of software objects designed to support sets of related functions, like sorting, random-number generation, and database searches

Debug format - The layout of debugging information within an object file format. Debug formats include stabs, COFF, DWARF, and DWARF 2.

Debug protocol - The mechanism by which a debugger examines and controls the program being debugged.

Debugger - A tool that allows programmers to examine and control a program, typically for the purpose of finding errors in the program.

Debugger - A tool used by a programmer to identify and fix bugs in a program.

DejaGNU/dejagnu - The regression testing framework based on tcl and expect.

Demarshalling - The reverse of marshalling (see marshalling).

Deterministic - When the time it takes to do something is finite and predictable is deterministic.

Development cycle - A typical development cycle involves writing source code, compiling source code into object modules, linking object modules into a binary executable and finally testing and debugging.

DII (Dynamic Invocation Interface) - DII defines the client's side of the interface that allows dynamic creation and invocation of requests to objects. The DII is another way to invoke a server's operations without using the stub. (See definition for stub.)

Driver - Software that is communicated between hardware peripherals and the rest of the system.

DWARF - A debugging format based on attribute records. Versions include DWARF 1, 1.1, 2, & extensions to 2.

E

ECOFF - Extended COFF, a format used with MIPS & Alpha processors, both for workstations & embedded uses.

eCos - Embedded Configurable Operating System, a complete, open-source run-time environment, allowing embedded system developers to focus on differentiating their products instead of developing, maintaining, or configuring proprietary, real-time kernels.

ELF - Extended Linker Format. Appeared with Unix SVR4 and used on many systems, including Solaris®/SunOS®, Irix®, and Linux®. Many embedded systems also use ELF.

Embedded system - Specialized computing devices that are not deployed as general purpose computers. An embedded system is preprogrammed to perform a narrow range of functions with minimal end user or operator intervention.

Endpoint - Endpoints are arguments that define some of the communication connections used by the components. Each argument is a comma-separated list of endpoints.

End-to-end predictability - In a fixed priority CORBA system, end-to-end predictability is defined as respecting thread priorities between client and server for resolving resource contention during the processing of CORBA invocations; bounding the duration of thread priority inversion during end-to-end processing' and bounding the latencies of operation invocations.

Exception handling - Event that occurs when a block of code reacts to a specific type of exception. If the exception is for an error from which the tool, the debugger for instance, can recover, the debugger resumes its process.

Exceptions - An unexpected event. The event can contain data, may be defined by the language, developer or the CORBA standard. An exception can be returned by the operation as an alternative. It is normally used to indicate an error condition.
Executable file - A binary-format file containing machine instructions in a ready-to-run form.

F

Fault tolerance - the ability of a middleware solution to operate (continue functioning) in an environment where elements have failed so that service is not interrupted.

Forward declaration - Forward declarations in IDL allow an interface to be referenced before it is declared.

G

Garbage collection - The automatic detection and freeing of memory that is no longer in use. A runtime system performs garbage collection so that programmers never explicitly free objects.

gas - Acronym for the GNU assembler. Interchangeably used with capitalization, as GAS.

gcc - Acronym for the GNU C compiler. Interchangeably used with capitalization, as GCC.

gcj - Front end to GCC that is able to read Java .class files, generating assembly code.

GDB - Main debugger used with GNU (command-line interface) The purpose of a debugger such as GDB is to allow you to see what is going on inside another program while it executes-or what another program was doing at the moment it crashed.

GDB standard remote protocol - An existing ROM monitor used as a GDB backend.

gdbserver - This is a control program for Unix-like systems, which allows you to connect your program with a remote GDB via 'target remote' - but without linking in the usual debugging stub. Gdbserver is not a complete replacement for the debugging stubs, as it requires essentially the same operating-system facilities that GDB itself does. In fact, a system that can run gdbserver to connect to a remote GDB could also run GDB locally!

gdbstub - Process and interfacing software to implement a protocol which is used for communication between the gdb debugger running on the host machine and the gdb debugger running on the target machine

GIOP (General Inter-ORB Protocol) - GIOP specifies a set of message formats and common data representations for communications between ORBs.

Glibc - A Standard compliant library that has been ported to a number of operating systems, and provides ANSI/ISO, POSIX, BSD and SystemV compatibility.

GNU - Recursive acronym for GNU's Not Unix. A project to build a free operating system, started by Richard Stallman in 1985, with many useful spinoffs, such as the Emacs text editor, a C compiler, a debugger, and many other programming tools.

GUI - Graphical User Interface, which refers to an interface and the techniques involved in using a keyboard or a mouse, for instance, to provide an easy-to-use interface to some software.

H

HAL - Hardware Abstraction Layer, which provides a portability layer to the rest of eCos so that higher layers do not need to be aware of the specifics of the architecture and platform. This layer is designed to be comparatively small and simple to implement, and is also component-orientated to allow sharing between related platforms. Also a famous series of intelligent computers depicted in movies.

Hard Real-Time - Where the schedulable entity must meet a hard deadline, and if not, the entity fails.

High availability - A measure of reliability which is the percentage of time the system is able to produce the required results.

Host - The computer system the developer works directly on

I

i386 - Typical name for the 32-bit members of the Intel x86 family. Members include 386, 486, Pentium ("i586"), and Pentium Pro ("i686").

IDE - Integrated Development Environment, a GUI tool or a set of tools that uses GUI functionality.

IDL (Interface Definition Language) - Language that specifies the interfaces of objects independent of any programming language specified by the OMG.

IIOP (Internet Inter-ORB Protocol) - IIOP specifies how GIOP messages are exchanged over a TCP/IP network.

.impl - .impl on the end of a class name means that it is an implementation of the IDL interface.

Incarnating - Incarnating occurs when a C++ class provides a programming language body for a virtual CORBA object.

Inheritance - The components that another object inherits. Objects inherit only operations and attributes. Multiple inheritance occurs when an interface inherits from more than one interface.

Instance - An instance is the representation of an object implementation being executed.

Interface - The interface is a collection of operations that provide services common to object implementations and clients. This includes initializing the ORB and obtaining object references.

IOR (Interoperable Object Reference) - An Interoperable Object Reference is the equivalent of a distributed, network smart, location transparent pointer.

Isochronal - A form of data transmission that guarantees to provide a certain minimum data rate, as required for time-dependent data such as video or audio. Isochronous transmission transmits asynchronous data over a synchronous data link so that individual characters are only separated by a whole number of bit-length interval. This is in contrast transmission, in which the characters may be separated by arbitrary intervals, and with synchronous transmission.

J

JavaTM - An object-oriented, "write once, run anywhere" programming language, developed by Sun Microsystems®, Inc.

JDKTM - A software development environment for writing applets and applications in the Java™ programming language, developed by Sun Microsystems®, Inc..

JIT - Just-in-time compiler that converts all of the bytecode into native machine code just as a Java program is run, resulting in run-time speed improvements over code interpreted by a Java Virtual Machine (JVM).

JTAG - Joint Test Advisory Group, referring to a type of hardware interface that allows the testing of chips and boards within a complete system; programs running on processors with JTAG support may be controlled through the processor's JTAG port.

JVM - Java Virtual Machine, part of the Java Runtime Environment responsible for interpreting Java bytecodes.

K

Kernel – Core component of an operating system.

L

ld - The GNU linker. Interchangably used with capitalization, as LD. See linker.

libgloss - The library for GNU Low-level OS Support, contains the startup code, the I/O support for gcc and newlib (the C library), and the target board support packages to which you need to port the GNU tools for an embedded execution target.

Library - A Collection of subroutines.

Linker - A tool that merges object files and library archives (such as compiled classes), building an executable, a complete program or a single executable file. For GNUPro Toolkit, the linker is the ld tool.

Linux - A free Unix operating system for many kinds of computers, created by Linus Torvalds and friends starting about 1990 (the pronunciation of /"lee-nuhks"/ is preferred, accenting the first syllable, since the name Linus has an /ee/ sound in Swedish).

Load balancing - The ability of distributed applications to disseminate work-loads as needed.

Location transparency - The client does not know whether the target object is local to its own address space, is implemented in a different process on the same machine, or is implemented in a process on a different machine.

M

Marshalling - Marshalling is the way the ORB converts a request or a reply into a platform independent data form that can be transferred across the network.

Message - A user-definable structure containing data being passed between processes.

Message queue - An ordered list of data or messages.

Middleware - Middleware creates the illusion of multiple servers behaving as one computer system. Alternatively, software that allows more than one application to share information seamlessly.

Module - A module is a group of interfaces and provides a name space for a group of objects. You can collectively refer to objects enclosed in a module.

MOM - Message Oriented Middleware is a client/server infrastructure that increases the interoperability, portability, and flexibility of an application by allowing the application to be distributed over multiple heterogeneous platforms.

Multithread - A thread is an activity that is executed sequentially. When more that one thread is executed, the activities are collectively referred to as multithreaded.

N

Naming Service - The Naming Service is simply a CORBA server with a standard defined interface to man names to IORs. The CORBA specification provides a Resolve_Initial_References as a short cut for obtaining a reference to the naming service. It also allows a way to map easily recognizable names to IORs.

O

Object - In object-oriented design or programming, a data item with instructions for the operations to be performed on it.

Object Adapter - The Object Adapter is the part of CORBA that defines how to activate a server so it can be used. It provides the server with access to the ORB and generates object references and invokes methods.

Object file - A binary-format file containing machine instructions and possibly symbolic relocation information. Typically produced by an assembler.

Object file format - The layout of object files and executable files. Common formats include a.out, COFF, & ELF

Object ID - A user or system specified identifier used to "name" an object within the scope of its Object Adapter. Object IDs are not guaranteed to be globally unique, not or they necessarily unique within a single server process. The only constraint is that each is unique within the Object Adapter where it is created and registered.

Object Implementation - An object implementation, also called a server, provides a response to requests for services for other objects, components, or applications. A given object, component, or application can be a server for some requests and a client for other requests.

Object Management Group (OMG) - The OMG is a non-profit consortium created in 1989 to promote the theory and practice of object technology for the development for distributed operating systems. The goal is to provide a common architectural framework for object-oriented applications based on widely available interface specifications.

Object Map - A table maintained by an object adapter that maps its active CORBA objects to their associated servants. Active CORBA objects are named in the map via Object Ids.

Object Oriented - having to do with or making use of objects; an object in this sense is a component containing both data and instructions for the operations to be performed on that data. In object-oriented programming, these reusable components are linked together in various ways to create applications.

Object Reference - An object reference contains all the information about where an object is located and what is needed to communicate with it.

Operation - An operation is an action that an object performs - the services that clients can invoke. Operations in IDL define a (possibly remote) call that can be made on an object and have zero or more arguments and return values. The definition of the arguments and return values in IDL is called the operation's signature. Operations can only be specified in IDL.

ORB (Object Request Broker) - The ORB communicates requests. It is responsible for identifying and locating objects, handling connections, and delivering data. (see CORBA)

OS (Operating System) - Software that controls the computer's hardware and determines what actions are possible from programs.

OSF/1 - The Open Software Foundation's version of Unix, used in Digital's Alpha machines.

P

Patch - A change in source code to correct or enhance processes.

PE - Portable Executable, Microsoft's object file format for Windows® 95 and Windows® NT operating systems. It is basically COFF with additional header information.

Platform - A specific implementation of a hardware or software architecture.

POA - Portable Object Adapter. The POA provides fundamental services such as object creation, servant registration, and request dispatching. It allows developers to write fully scalable, high performance server applications.

Process - An operating system provided facility for executing a program in a separate address space. Usually executed in parallel with other processes via time slicing or interrupt driven priorities.

PROM - Programmable Read-Only Memory, ROM that can be programmed using special equipment. PROMs can be programmed only once. Compare with EPROM.

Q

QoS - Quality of Service

R

RAM - Random-Access Memory, referring to volatile memory that can be read and written by a microprocessor.

RDP - Remote Debugging Protocol, a protocol used with ARM's Demon monitor.

Real-time system - The ability to respond to events predictably and on-time.

RedBoot - A standardized (by RedHat?) embedded debug and bootstrap solution based on the eCos HAL, that provides firmware for running and debugging eCos, GNUPro applications and embedded Linux systems on a wide range of embedded platforms. RedBoot supports remote, portable downloading and debugging over a variety of media, flash and network booting, and downloading and updating of flash images remotely via serial or ethernet connections for field upgradeability.

Reference - A reference is a pointer in C++.

Referencing Domain - The Referencing Domain is simply a developer-defined string and represents the developer's partitioning area. This referencing domain

is set by an environment variable and is also passed to the daemons through a command line argument.

Registers - Registers are settings representing values that serve as temporary storage devices in a processor, allowing for faster access to data. Registers are divided into several classes: pseudo registers, temporary registers, and machine registers.

Remote target - See target.

RISC - Reduced Instruction Set Computer, machines typically having fixed-length instructions, limited addressing modes, many registers, and visible pipelines. Examples include MIPS, ARM, SH, PowerPC, and StrongARM.

RMI - RIFF MIDI File (file name extension); RIFF or Resource Interchange File System is a multimedia file format, the Microsoft equivalent of Amiga IFF format

ROM - Read-Only Memory, non-volatile memory that can be read, but not written, by the microprocessor

RTOS (Real-Time Operating System) - An operating system designed for real-time systems.

S

SAP (Service Access Point) - The OSI term for the component of a network address which identifies the individual application on a host that is sending or receiving a packet.

Servant - A servant is a programming language entity that implements one or more CORBA objects. Servants exist within the contexts of a server application. In C++, servants are object instances of a particular class.

Server - A server (also known -sometimes- as an object implementation) provides a response to requests for services from other objects, components, or applications. An object, component, or application can be a client for some requests and a server for other requests.

Skeleton - The server skeleton is generated by the IDL compiler. The server code is how the server operations are accessed.

Soft Real-Time - Schedulable entities don't meet time constraints and may miss a deadline. Upper bounds are soft, and the predictability of non-schedulable entity timeliness is sub-optimal

Solaris® - Sun's current version of Unix, superseding SunOS. Based on SVR4 Unix. Sun officially calls it SunOS 5.x, with versions including 2.0 through 2.6 (or, as Sun refers to them, 5.0 through 5.6).

SPARC - Name for the family of RISC processors based on Sun's SPARC architecture. Members include SPARClite, SPARClet, UltraSPARC, SPARC v7, SPARC v8, and SPARC v9.

Speed of response generalizes systems as fast or slow**,** with no clear boundary between the two. Any system with sub-second responses is fast and any system with response times in minutes is slow, leaving a gray area of seconds.

stabs - Based on symbol tables, a debug format originally introduced with the Berkeley Unix system, which records debugging information in certain symbols in the object file's symbol table. stabs information may also be encapsulated in COFF or ELF files.

Stub - A small piece of code that executes on the target and communicates with the debugger, acting as its agent, collecting registers, setting memory values, etc. Also, in a native shared library system, the part of the shared library that actually gets linked with a program.

Stub - The client stub is generated by the IDL compiler. This piece of code allows the client to invoke an implementation object's services.

Synchronous - a means of sending data in which a clocking signal is used and the characters are separated by time intervals, rather than by start and stop bits as in asynchronous transmission.

T

Target - A computer system the developer is writing code for. This may or may not be the same platform as the host. Used both to refer to an actual physical device, and to the class of devices.

Task - In a multi-tasking environment, an independently running program or subprogram. Each task is assigned a task number. Another term is thread.

TCP/IP - Transmission Control Protocol based on IP. This is an internet protocol that provides for the reliable delivery of streams of data across the web.

telnet - Standard terminal emulation protocol in the TCP/IP protocol stack, used for remote terminal connection, enabling users to log in to remote systems, thereby using resources as if connected to a local system.

Thread - A thread is an activity that is executed sequentially. More that one thread can be run concurrently. Another term for thread is task.

Toolchain - Informal term for the collection of programs that make up a complete set of cross-compilation tools. Typically consists of the following example's sequence:

compiler->assembler->archiver->linker->debugger

Transport - communications mechanism transporting data between client and server.

Typedef - In IDL, the typedef statement provides the ability to define a new name for an existing IDL type (basic or composite). This new name is merely a renaming and does not constitute a different type.

U

UML (Unified Modeling Language) - A non-proprietary, third generation modeling language. The Unified Modeling Language is an open method used to specify, visualize, construct and document the artifacts of an object-oriented software-intensive system under development. The UML represents a compilation of "best engineering practices" which have proven successful in modeling large, complex systems.

Unix - Unix operating system. The uppercase spelling of 'UNIX' is used interchangeably. Invented in 1969 by Ken Thompson after Bell Labs left the Multics project, Unix subsequently underwent mutations and expansions at the hands of many different people, resulting in a uniquely flexible and developer-friendly environment. By 1991, Unix had become the most widely used multiuser general-purpose operating system in the world.

V

VFS - Virtual File System architecture.

Virtual machine (VM) - An abstract specification for a computing device that can be implemented in different ways, in software or hardware. Compiling to the instruction set of a virtual machine is much like compiling to the instruction set of a microprocessor, using a bytecode instruction set, a set of registers, a stack, a garbage-collected heap, and an area for storing methods.

VxWorks – A Real-time operating system from WindRiver. Well known after its travel to Mars to control the Sojourner computing resources.

W

Win32 – A well known API from Redmond.

X

X - An allegedly "over-sized, over-featured, over-engineered and incredibly over-complicated" window system developed at MIT and widely used on Unix systems. With its sources freely available, it is a vehicle that is widespread since developers can modify and customize it according to their requirements.

x86 - A generic name for the Intel 8086 architecture family.

XCOFF - eXtended COFF, IBM's object file format for RS/6000 and PowerPC systems.

XML - Extensible Markup Language. An initiative from the W3C defining an "extremely simple" dialect of SGML suitable for use on the World-Wide Web.

Y

YACC – Yet Another Compiler Compiler.

YAST – Configuration tool for Linux systems.

Z