IST-2001-37652

Hard Real-time CORBA

Title

RCT Requirements Specification

Authors

Klas Nilsson (klas@cs.lth.se)

Reference	IST37652/0439
Date	2002-01-03
Release	0.95
Status	Draft
Clearance	Consortium

Partners

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

Summary Sheet

IST Project 2001-37652
HRTC
Hard Real-time CORBA

RCT Requirements Specification

Abstract:

Proposed extensions of real-time CORBA to meet hard deadlines should be confronted with relevant and demanding test cases. One such very demanding application area, reflecting common industrial requirements, is robot control. Robots in general are required to be quite flexible, and therefore the flexibility of a (CORBA-based) component-oriented design is particularly useful. For industrial robots in particular, efficiency and predictability need to be carefully considered, both in terms of computing and communication. These aspects apply to both low-level control with sampling frequencies well above 1 kHz, and up to non-real-time levels of control and user interaction. On all levels there is a need to be able to connect external sensors providing real-time data for the feedback control, based on application / customer needs and using the flexibility of Hard RT CORBA. Such a Robot Control Testbed (RCT) is specified to this end.

Copyright

HRTC Partners:

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

Release Sheet (1)

Release:	**0.1 Internal Draft**
Date:	2002-09-23
Scope	Initial outline
Sheets	All

Release:	**0.2 Draft**
Date:	2002-12-23
Scope	Application requirements completed
Sheets	All

Release:	**0.9 Release Candidate**
Date:	2002-12-29
Scope	Initial version
Sheets	All

Release:	**0.95 First Draft Release**
Date:	2003-01-03
Scope	Figures and references completed
Sheets	All

Table of Contents

1 Introduction 6

1.1 CORBA considerations 6

1.2 Characteristics of the application 7

1.3 Testbed approach 9

2 CORBA for Flexible Manufacturing Systems 10

3 RCT Requirements 12

3.1 Specific requirements 12

3.1.1 System monitoring and task submission 12

3.1.2 On-line connection of objects for engineering and computing 12

3.1.3 Deployment of external sensing and control 13

3.1.4 Application specific control using distributed sensing 13

3.1.5 Configuration of built-in manipulator control 14

3.1.6 Distributed HRT control of robot motions 14

3.2 Concluding remarks 15

4 Reference list 15

1 Introduction

The objective of Work Package 3 is the construction of a test application in the field of robot control (RCT: Robot Control Testbed). The limitations of current CORBA implementations and standards should be possible to verify experimentally, in a way that is meaningful from an application point of view.

1.1 CORBA considerations

The purpose of CORBA is to enable communication among application clients and object implementations, in a simple and language neutral manner. The standard CORBA facilitates this by uniformly defined RPC calls to services, common principles of naming, error handling, and by supporting integration with legacy application code. The Object Request Broker (ORB) plays a central role in the separation of client interfaces and (typically remote) object implementations, by managing the lookup and use of distributed objects on a common network. Typically this is accomplished using the TCP/IP type of transport and the Internet Inter-ORP Protocol (IIOP) for the interaction between ORBs, but without supporting bounded-time calls and resource management. For these and other reasons, standard CORBA is not believed to be useful for embedded control systems, but that should be possible to illustrate in the RCT.

To improve on predictability and resource control for distributed concurrent applications, the RT CORBA has been defined. Instead of constraining CORBA, RT CORBA extends the specification in such a way that the RT application developer is given a set of features for improvement of end-to-end predictability. This involves the functionality of the ORB and some additional interfaces. The RTOS and its scheduling is assumed to be POSIX compliant, meaning that priorities are considered strictly and priority inversion times are bounded. The communication transport, however, is assumed to work as in the standard CORBA. This normally means TCP/IP, and despite the well-known unpredictability of that protocol, there is no appropriate interface for dealing with communication deficiencies. The RCT must pay careful attention to the deployment of networking suitable for feedback control.

Threads scheduled to run on a processor and messages scheduled/transmitted for transport according to a communication protocol can be referred to as activities. For HRT CORBA, activities must naturally be of primary concern. In RT CORBA on the other hand, the pure operation-oriented view of objects is maintained (see http://doc.ece.uci.edu/CORBA/formal/02-08-02.pdf):

> Real-Time CORBA does not define IDL for an activity. Instead of worrying about how to delimit an individual activity, it deals with invocations of IDL defined operations.

These are well-formed concepts in the OMA. An operation invocation consists of a Request and a Reply. It is initiated by some client computational context; for example, a thread and passes through a client-role ORB, a transport protocol (TCP in the case of GIOP), a server-role ORB (possibly involving queuing) to a server application. Thereafter the operation passes through the same entities in reverse order, back to the client. An activity may encompass several, possibly nested, operation invocations.

This specification acknowledges that an abstract activity is represented by concrete entities: a message within a transport protocol, a request held in memory, and a thread scheduled to run on a processor. These three phases are termed “in-transit,” “static,” and “active” respectively. Real-Time CORBA provides the ability to effect these three phases of an activity. It leaves the developer to delimit their concept of an activity by the way they coordinate these concrete entities using the interfaces specified.

HRT CORBA has to go beyond RT CORBA in that handling and completion of activities have to be explicitly supported in order to support predictable feedback control. This includes controllability of any available predictable transport protocol/network, and an IDL more appropriate for dealing with performance and Quality of Control (QoC). It is a primary issue for the RCT to facilitate experiments for verification of proposed HRT CORBA features in these areas.

1.2 Characteristics of the application

Control of industrial robots can be characterized by the combined need for flexibility (with respect to new tasks and unforeseen application demands) and the need for performance (to accomplish productivity and profitability). We may distinguish, in any robot controller more or less explicitly, three levels of control (see Figure 1) with different requirements also from a CORBA point of view.

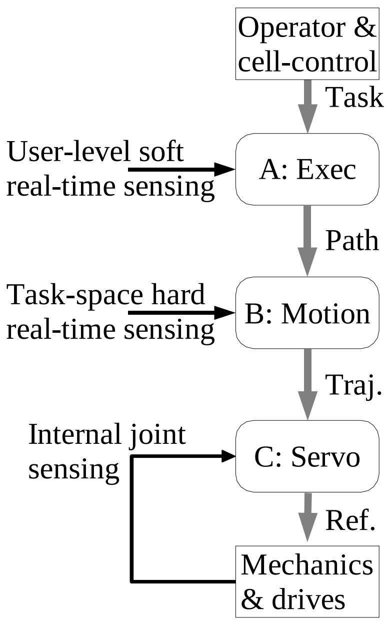

Figure 1 Robot control levels with sensor input.

A. End–user level providing task specification using robot programming languages / tools, or host computer interfaces to obtain CAD data and the like. Real-time requirements are soft since robot controllers are designed such that increased response times simply result in the robot working slower, but still performing its task unless the delays are too severe. Due to buffering and asynchronous communication with the motion control, some sporadic delays (sometimes up to one second) can be accepted. Host computer communication may even be delayed several seconds in standard systems today, since communication is for configurations and task descriptions that do not effect the robot while it is working. However, there is a desire to be able to use standard PC-based *sensors and on-line changes of motion specifications from supervisory systems*. The RT requirements then is hard or soft with required response times in the range of 50 to 500 ms.

B. Generation of trajectories and control parameters for motion control of the manipulator(s) take place on an intermediate level. Here sensors used for *application or task-specific feedback to the motion control* are also connected, locally via dedicated interfaces or field buses. This type of sensing is today limited to some predefined types such as laser scanners for seam tracking in arc welding, as provided by the robot manufacturer. Depending on the type of the sensor and the required feedback, sensor readings need to be done periodically with a period of 1 to 100 ms. There is a strong desire within advanced robotics to permit customers to add real-time components for this type of application specific control, typically in Cartesian space using vision and force sensing. Real-time requirements are hard, but in many cases occasional missed deadlines can be tolerated. With multiple sensors and limited computing and communication resources, some *Quality-of-Control* (QoC) measure in combination with control components are desirable for convenient tuning of productivity.

C. *Built in low-level control of manipulator motions* is mainly designed by the developer of the embedded robot controller. The trend here is to use distributed control via a hard real-time communication network, with nodes for joint sensing and motor drives, possibly including multiple arms. Hard real-time means very few missed communication / control deadlines, say 1 per 1000 which will not influence motion performance or accuracy. Fully deterministic and dependable communication will, however, simplify system design and implementation. It is desirable to be able to use software components for the control, but very little overhead is acceptable due to sampling periods which vary between 0.1 and 10 ms, depending on dynamics (size of the robot) and performance requirements.

For the top-level control, including cell control and integration with PLC systems, the situation resembles that of the Process Control Testbed (PCT); please refer to that specification for those aspects. The motivation for the RCT is more on the items B and C: Feedback control requiring high bandwidth, both in terms of control performance and the needed data communication in the case of distributed sensing.

Since the built-in control (item C) is structurally know at boot time, it is acceptable to statically configure the system off line. In that configuration, certain unused resources (such as scheduled communication slots) can be made available for the on-line flexibility, if application specific features are to make use of built-in resources (e.g. using the internal drive-sensor communication also for customer sensors on the robot end effecter).

1.3 Testbed approach

Construction of the testbed is to include a distributed robot control application using CORBA technology developed in the HRTC project. Since the time for development of the testbed is quite limited, earlier experiences and platforms have to be utilized. The robot systems available to this project include three industrial robot systems including standard ABB industrial manipulators:

- The system of primary interest is based on an IRB-2000 / S3 robot that has been completely reconfigured for control experiments. Distributed computing located with internal sensors and drives is based on Axis

ETRAX computers, while the central control computers are of Motorola PowerPC type. These computers need to communicate via some hard real-time transport. Currently this means scheduled and switched Ethernet, but a dependable transport such as the TTP/C would be an attractive alternative if latency and bandwidth demands were fulfilled.

- As a simplified (from a control and complexity point of view) platform, one IRB-6/S2 robot is available. Due to its simple DC motor drives and sensor electronics co-located with the drives, the control level C (see items in previous section) is comprised by only one node. External sensing and higher levels of control is similar to the IRB-2000 system, and in both these systems the complete software is open to changes for whatever is needed in the testbed.
- As a representative for interfacing with legacy or native systems, the most recent type of ABB robot controller, an IRB-2400/S4C+ system, is available. Only some additional interfaces in terms of callback hooks, shared memory areas, and outgoing interrupts, have been added. The original software system (running on four embedded CPUs) remains and provides extensive robot programming support. A component based design of the IRB-2000 system with appropriate interfaces should apply to the IRB-2400 as well. This would be interesting to test mainly on the levels A and B (programming and external sensing)

Host computers providing engineering platforms and operator interfaces run on PC (Win32 and Linux), Sun, and SGI workstations. This is ideal for experimenting with interconnection of heterogeneous systems, but the main development will (for reasons of stability and control) be based on Linux. The different embedded processors mostly use homemade RT kernels with dedicated but not quite general networking support. For improved availability of the testbed, these embedded nodes should preferably be Linux-based, with RT extensions such as the RTAI for meeting hard RT deadlines.

In addition to the real physical testbed, a virtual testbed capturing some of the main aspects is also needed for experimenting at different sites and for portable demonstrations.

2 CORBA for Flexible Manufacturing Systems

The situation today is that the integration of subsystems within manufacturing is hampered by the variety of systems and interconnection principles. Software cannot simply be relocated to operate on a uniform platform such as Java or Microsoft .NET. Machine controls require special run-time properties, and factory control includes proprietary applications that are not open to source code changes for the purpose of integration. Instead, proprietary or legacy applications need to be integrated by using a distributed object model such as CORBA, but RT requirements need to be carefully considered.

Data exchange between different sub-systems is usually done via files, in a variety of vendor specific formats. Direct connections via well-defined object interfaces, where objects encapsulate controllers and host applications, would in most cases be a better alternative. When direct connections are used, it is mainly via DDE or socket-based communication over TCP/IP. In both these cases, a client-server model requires the server to be started first and then the clients can connect. For simple point-to-point connections this is acceptable. For more complex situations, such as sensor-based robot motions and advanced cell control utilizing host computer applications, the situation is different; the simple client-server model results in complex and fragile startup sequences. Also recovery from networking errors is problematic, adding unnecessary complexity from an operator point of view.

Looking at the development of machine controls, the situation is similar but problems mainly stay within one company. Reuse of control software is mainly limited to the use of function libraries, which are then called from manually tailored control classes and objects. A component-based approach for more extensive reuse of control software is desirable, but such techniques have not been accepted / useful in industry so far, but the development is promising. OROCOS is an ongoing open source software project aiming at creating highly configurable control components, based on CORBA principles. However, CORBA is used mainly for the IDL and component interfaces, while distributed communication and RT properties of the run-time system is left out of the IDL. Instead, timing considerations are managed by supporting mechanisms in the control system architecture, but without HRT communication and execution. We consider the OROCOS effort to be complementary to the HRTC project.

Communications in manufacturing systems take place both horizontally between units on the same level of the overall system, and vertically between units on different hierarchical levels. Horizontal communication within the factory or line level communication does not need to be in real-time; ordinary CORBA and TCP/IP works well enough. On a cell control level, communication needs to be RT but in most cases without hard deadlines. For the machine and robot control, RT demands are as mentioned in principle hard. Vertical integration can include communication between objects used in HRT parts of the system and objects that do not have to explicitly consider timing. Nevertheless, such vertical calls (e.g. obtaining an embedded sensor or control value to be used for the high level job scheduling) are crucial for the total integration of modern production systems.

Hence, it is important to maintain interoperability with (non-RT) CORBA for the vertical integration of manufacturing systems, but to really be useful integration must include the real-time machine control level, which in turn clearly requires CORBA to be RT enabled.

3 RCT Requirements

The key issue in the RCT is to facilitate experimentation of distributed robot control with feedback through the net, both using (non-RT) CORBA, (soft-)RT CORBA, and HRT CORBA. Ranging from a top enterprise level view of the system (for submission of manufacturing tasks and monitoring of the manufacturing process) and down to the HRT control of sensor-based manipulator motions, the different types of requirements are now to be listed.

3.1 Specific requirements

The specific requirements will now be listed for each separate area of interest. Note that only the final specification (#6) is required for the HRTC project; the others are listed for completeness and as a plan for further work, part of it hopefully within the HRTC project.

3.1.1 System monitoring and task submission

Robot tasks can be defined either manually on-line via some teach pendant, or off-line using some programming tool (typically utilizing CAD data). Consider the off-line case, which is most interesting from a distributed and enterprise point of view. Seen from the host computer, the robot system accepts configurations, robot programs, and changes of coordinates in already loaded programs (even when the robot is performing the task defined by the program). In a similar way, the status of the system can be retrieved, for instance by the host before deciding about submission of new tasks.

Different robot vendors provide more or less sophisticated network interfaces. The ABB S4C+ controller provides an RPC-based interface called RAP (Robot Application Protocol). Standard use does not require any real-time response, but one-line control (see item A above) by real-time coordinate changes illustrates the need for real-time performance even if deadlines are modest (are 200 ms for the IRB-2400). Encapsulating the RAP interface in CORBA would be useful for RT CORBA experiments.

3.1.2 On-line connection of objects for engineering and computing

The computations normally needed for the command and control of a manipulator is implemented in the embedded real-time controller that is part of the robot system. For customer specific feedback from external sensors, additional computing hardware may need to be added. However, for the same reasons that CORBA is useful for enterprise systems, software applications available on host computer system can be crucial to support embedded computing (such as optimal control or motion planning). For an established application area all computations can of course be ported to the embedded real-time system. But for prototyping and special solutions, the availability of distributed objects implementing part of the system can then be what makes the development feasible at all. The testbed should permit such experiments to be carried out.

3.1.3 Deployment of external sensing and control

An open robot control system provides slots in which additional control objects can be installed, with access to an embedded context defining the interface to the enclosing system. Such an interface can of course be described using the CORBA IDL, but what about the control objects to be installed? Can such an object also be self contained and useful also outside the embedded context, e.g. in a simulation environment for virtual manufacturing?

Customer specific control objects can be written by hand or generated from some kind of high-level description. The latter alternative has proven to be very useful, given state-of-the-art code generators from block diagrams (Simulink Real-Time Workshop and Embedded Coder) or from model libraries (e.g., in Modelica). The benefit from this approach is simply considered as a matter of engineering productivity, since it is quicker to edit a graphical description than to write and debug source code. However, equally important but never (?) mentioned is the improvement in composability; assuming correctness of the comparably small building blocks written in a well defined framework, a composed hierarchy of control blocks or controllers stays (ensured by tools such as Real-Time Workshop) within the 'sandboxes' of the used blocks. This results in composability and thereby also scalability, opposed to manual composition of controllers in C/C++ that does not impose the needed restrictions on the (perhaps inexperienced) programmer.

The role of HRT CORBA in this perspective is to encapsulate controllers generated in this way, to make them callable in a distributed environment, and to facilitate the deployment in the embedded controller. The deployment stage makes this item different from the others; cross compilation and dynamic loading into embedded targets should be supported. Hence, the testbed should explicitly permit such experiments to be carried out.

3.1.4 Application specific control using distributed sensing

With the customer or application specific control deployed manually or according to the previous item, the added control features (such as visual servoing or force controlled motions) should provide feedback from the partly unknown environment to accomplish a more skilled and productive robot. Assuming the built-in control of the manipulator as fixed and proprietary (as for the IRB-2400 mentioned above), multiple sensors still provide a great deal of flexibility. Often there is a trade-off between different sensors, in particular when communication and computing resources are shared. For instance both force and visual feedback can be used to identify the location of an object, but what is the combination that gives the best QoC? The testbed should provide a platform for such experiments with QoC measures in CORBA terms.

3.1.5 Configuration of built-in manipulator control

The built-in control of manipulator motions is, as mentioned, statically configured. In systems today, this is done in a fixed way by the robot manufacturer, using communication and computing resources that are explicitly available for the internal proprietary motion control. For future flexible and efficient sensor-based control, this will need to change. Modern control systems already are distributed, with local processing co-located with the sensors and the actuators. With a more flexible and open (to the customers), although static, structure of the control, it would be possible to let the built-in control share the real-time communication transport with customer sensing. This is not only a matter of cost; physically protected cabling inside manipulator arms is often in practice not possible to add, hence sharing the internal communication is highly desirable. Since internal communication is statically designed to meet hard deadlines, the testbed should preferably be useful for such configuration experiments.

3.1.6 Distributed HRT control of robot motions

With the above specifications #1 to #3 more or less fulfilled by the testbed, and combining the features for specification #4 and #5, the major purpose of the testbed is to facilitate evaluation of HRT CORBA from a control point of view. This included both statically defined (built in) control loops and dynamically defined (external) control loops. The influence of more or less deterministic communication should be possible to evaluate. The focus is on the motion control as such; the PCT captures the aspects of the overall factory control.

The built-in control should include servo control of at least six robot joints with a minimum sampling frequency of 4 kHz, but preferably 8 kHz or higher should be possible. The nominal delay from sensing to actuation for each joint may not exceed 1 sampling period. The internal control should handle torque, speed, and position control of the six robot joints.

The external control should influence the position setpoints of the internal control, as well as feedforward terms to joint speed and torque if necessary for relevant performance. The external sensors should be encapsulated as HRT CORBA objects. The sensor data transport should be accomplished via TCP / IP, UDP / IP, and via some deterministic scheduled communication.

For both the internal and the external control, the influence of jitter and lost samples should be possible to evaluate. A simulator capturing the communication aspects of control should be made available.

3.2 Concluding remarks

It is noteworthy to see that HRT CORBA does not exist isolated in its own domain; the interconnection with standard CORBA objects needs to be preserved. It is, however, the Distributed HRT control of robot motions that comprises the minimum scope of the RCT, permitting performance evaluation of proposed standards. The other items are desired for a more complete illustration of CORBA technologies, and the aim of the RCT is to provide an environment suitable for experiments in all these areas.

4 Reference list

[RTCORB] OMG: Real-Time CORBA 1.0 Specification.

[InCORB] Mowbray, T. J. and W. A. Ruh: “*Inside CORBA*”, Addison Wesley, 1997.

[Nil99] K.Nilsson and R.Johansson: “*Integrated architecture for industrial robot programming and control*”, Robotics and Autonomous Systems, 29, 205-226, 1999.