# WRAPPING A MOBILE ROBOT WITH RT-CORBA

**Ricardo Sanz** [*,1] **Adolfo Hernando** [*]
**Carlos Martínez** [*] **Ignacio López** [*]

[*] *Autonomous Systems Laboratory*
*Universidad Politécnica de Madrid, Spain*

Abstract: Building complex controllers is a major challenge and it is widely accepted that object technology can help with the problem. This is of special relevance in the field of complex robot control, in particular when distribution is necessary. CORBA is a suitable technology for deployment and is well demonstrated in the experimental field and in commercial robots. In this paper we describe the implementation of a real-time object wrapper for a mobile robot using Real-time CORBA technology. This type of wrapping enables the implementation of networked robot control systems with increased degrees of predictability.

Keywords: Robot control software, distributed control, object wrappers, CORBA, Real-time CORBA.

## 1. INTRODUCTION

The nature of applied research in intelligent robot controllers makes having a versatile software architecture a real need for exploring alternative designs in robotic mind construction. Flexibility, modularity, maintainability and even hot-replaceability of components are major non-functional needs for such systems. While some effort has been spent on genericity is robot control system construction, most research has been centered around the provision of ultimate architectures and reusable software components to fulfill specific missions in the robot controller. Less effort has been put, however, in the development of a robust and flexible underlying software platform where to explore such designs and components.

However, as it can be seen from the research on intelligent robot control, it is not so easy to provide an ultimate robot control architecture nor a general factoring of controller functionality as to provide true generic software components for specific roles into the architecture. The approach we pursue in our ASys project can be regarded as a *product line* approach(Sanz *et al.*, 1999*b*). We try to explore alternative control designs —the product line— as a collection of implementations that share a set of common assets —i.e. as a product family. Building product lines as product families is a major reengineering objective for development processes.

The particular underlying technology that we have selected for this research is the technology of modular, distributed, embedded objects provided by the CORBA suite of specifications. This paper presents an experiment in the line of providing distributed implementations over such an framework architecture based on these well known interoperability standards.

This paper demonstrates not only the feasibility but the convenience of using state-of-the-art, modular software technologies for the construc-

[1] Partially supported by the European Comission through project IST COMPARE.

tion of advanced robot controllers. This is the only possibility to tackle the complexity requiered for the controller of autonomous systems(Huang *et al.*, 2003).

## 2. DISTRIBUTED SOFTWARE FOR CONTROL

### *2.1 Aspects of distributed computing*

Most robot controllers of a minimum complexity are *distributed systems*. A distributed computer system is defined to be a system of multiple autonomous processing elements, cooperating in a common purpose to achieve a common goal; in this case the control of the robot to perform its mission.

The reasons for the distribution are multiple but two are the most important in the robotics domain:

- Deployment constraints that requiere the use of a particular computing platform (e.g. to run a specific sensor driver or employ a particular computer-generated implementation of a software module)
- Performace requirements that force the use of several computers (e.g. for sensor processing or high level real-time planning)

The traditional approaches to distribution were based on the use of low level APIs for specific communication protocols or the use of libraries for parallel computing. This aproach, however, does not scale well to the complexity level that is requiered in modern autonomous robots.

The approach used in our research is the employment of real-time middleware to support the implementation of complex controllers as collections of interacting real time agents(Sanz *et al.*, 1999*a*). Many types of middleware are in use today like Transaction Processing Monitors (TPM), Remote Procedure Calling (RPC), Message Oriented Middleware (MOM) or Object Oriented Middleware (OOM) (see Table 1)

Obviously, component reusability can only be achieved if the components do address domain needs and have well known and public specifications of provided and requiered interfaces (Radermacher *et al.*, 2005). This has already been recognised by the robotics community and a DSIG has been created inside the OMG [2] and an Request For Information has been publised. The OMG Robotics RFI(OMG Robotics DSIG, 2005)

[2] The purpose of the OMG Robotics DSIG is to foster the integration of robotics systems from modular components through the adoption of OMG standards.

| | |
|---|---|
| TPM | IBM CICS |
| | BEA Tuxedo |
| | HP Encina |
| RPC | Gradient RPC |
| | DCE de Compaq (HP) |
| | RPC de Sun |
| MOM | IBM MQSeries |
| | Software AG EntireX |
| | TIBCO Enterprise Message Service / Rendezvous |
| OOM | Microsoft COM/DCOM |
| | RMI de Sun |
| | CORBA del OMG |

Table 1. Some middleware examples classified by category. Only RPC and OOM seem relevant for control system loop implementation.

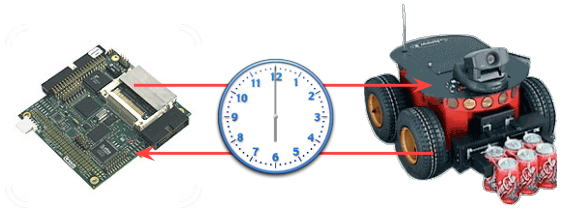

Fig. 1. Timeliness of networked interaction is critical for the performace of a distributed control loop.

seeks information which will be used to direct future standardization efforts in the area of reusability and interoperability of robotics technology.

### *2.2 Networked control systems*

A specilly interesting distributed system of relevance to our investigation is the networked controller(Hristu and Levine, 2005). In such a controller a control engine runs in a computer while the sensor and/or the actuator are run by software in other computer connected by a network. Different end-to-end properties and guarantees —timeliness in particular— can be provided by different OSs and networking infrastructures.

In our approach we try to achieve a unified technology implementation of any complex distributed controller without sacrificing any feature that may be needed at any level. That is of special importance regarding predictability issues in networked controllers. In this way we can guarantee the temporal behavior of a distributed controller withouth worrying about the details of the different layers.

Within a node, tasks interfere with each other through pre-emption and blocking when accessing shared resources. The execution times of the tasks themselves may be data-dependent or may vary due to hardware features such as processor caches. On the distributed level, the comunications gives rise to delays that can be more or less deterministic depending on the communication protocol and

the hardware infraestructure. This will affect the level of performace of distributed activities (e.g. degradation of real-time coordination of a group of autonomous vehicles).

Some sources of communication delays are the time needed to pass the messages through the different protocol layers in the sender and receiver hosts and forward the messages to other nodes, wait in the output queue buffer, transmit the message into the link, propagate over the link, acknowledge and (possible) resend in reliable transport protocols and link-layer resend in case of collision detection. RT distributed platforms provide APIs to somewhat configure these inner mechanics and provide QoS control.

Providing a framework for isolation between the functional application level and the underlaying mechanisms for real-time performance is hence critical. RT middleware can provide part of this isolation while other parts are coming from the more emcompassing embedded components approach [3].

### *2.3 Frameworks for distributed controllers*

There are not many attempts to develop a frameworks with the required properties but some attempts are worth mentioning in robotics and related fields.

OROCOS consists of two decoupled but integrated sub-projects:

**Open Realtime Control Services** —This is a hard realtime software framework directed toward all possible machine control applications (an outreach of the project's original robotics focus). It is designed to run safely parallel user defined tasks, on Linux 2.6 and RTAI (for hard realtime). It provides some interesting features like hardware and operating system abstraction, event handling, hierarchical and parallel state machines, multiple time- and event-triggered threads, advanced data protection for synchronous/asynchronous data flow, strongly typed data flow, *etc.* Integration between RTAI and TAO is also part of the ongoing work towards a hard realtime distributed control infrastructure based on CORBA.

**Open Robot Control Software** —A set of class libraries and an application framework offering generic functionality for machine tools and robots: cascaded control loops and control components, motion generation and interpolation; kinematics and dynamics; robot-specific control algorithms; estimation and identification; *etc.*

[3] See, for example, www.ist-compare.org.

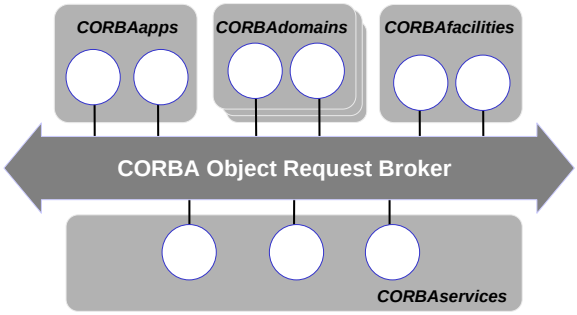


Fig. 2. CORBA technology offers resources at multiple levels for the construcion of complex distributed applications.

ORCA is an open-source suite of tools for developing component-based robotic systems. It provides the means for defining and developing components which can be pieced together to build arbitrarily complex robotic systems, from single vehicles to distributed sensor networks. In addition it provides a repository of pre-made components which can be used to quickly assemble a working robotic system.

## 3. CORBA CONTROLLERS

### *3.1 CORBA Technology*

We have selected the CORBA suite of specifications [4] for the purpose of serving as a base standardization platform where to build the required technology.

This collection of specifications includes the CORBA specification itself (Common Object Request Broker Architecture) but also the specifications related to the implementation on small footprint systems (i.e. the Minimum CORBA spec), predictability augmentation (the Real-time CORBA spec), fault tolerance (the Fault-tolerant CORBA spec) and componentization (the CORBA Component Model and the Lightweight CORBA Component Model).

Of special importance is the model proposed by the Object Management Architecture (see Figure 2) that specifies a categorisation of prebuilt components that enable the specification of standardized services of horizontal scope like the CORBA Facilities or CORBA services (time, persistence, evente, transactions, etc.) or of domain scope. This last case has got more interest with the creation of the Robotics DSIG inside the OMG [5].

CORBA technology —in particular object request brokers— has been used in many applications in robotics (specially in distributed robotics). Some past experiences in our laboratory are:

[4] Publicly available from `www.omg.org`.
[5] See `http://robotics.omg.org/`.

- A reimplemenation of the distributed interaction mechanisms in the ARCO multirobot systems(Sanz *et al.*, 2001*b*).
- An implementation of a CORBA server wrapping the local API of a mobile robot Pioneer 2AT (Pareja, 2004).

This last implementation served as a basis for the system described in this paper.

### *3.2 Real-time CORBA Technology*

The Real-time CORBA specification (Schmidt and Kuhns, 2000) was developed to provide further control to the application developer to improve application predictability. RT CORBA specifies aditional mechanisms to increase the control of resources to improve end-to-end predictability of distributed object applications. There are two main specifications for real-time CORBA: The 1.1 specification relies in an extension of key CORBA entities and services that allows managing three types of core resources: processor, memory and communication. The 2.0 specification incorporates mechanisms for dynamic scheduling.

Regarding increased predictability the RT CORBA specification defines the concept of end-to-end predictability(e.g. predictability of a networked control loop from sensor to actuator): respecting thread priorities between clients and servers during the processing of CORBA invocations and bounding latencies of operation invocations and priority inversions during end-to-end processing. This specification only supports fixed-priority scheduling whereas another specification has been created to address dynamic distributed systems. This second specification generalizes the concepts of distributed system scheduling and distributable thread in order to allow control on the scheduling discipline and associated parameter elements. The schedulable entity is now a distributable thread that may span node boundaries.

### *3.3 CORBA Control Systems*

CORBA has been used in many control applications with varying temporal requirements. Some examples from our laboratory are:

- Emergency management in chemical plants (Sanz *et al.*, 2000)
- Networked process control systems (Sanz *et al.*, 2005)
- Electrical substation protection (Sanz *et al.*, 2001*a*)
- Cooperating movile robots (Sanz *et al.*, 2001*b*)
- Strategic control of cement plants, etc.

See (Segarra, 2005) for more details on the use of CORBA technology in control systems.

## 4. RTHIGGS WRAPPER

### *4.1 The SOUL Project*

The SOUL project tries to develop a cognitive architecture for complex cognitive control of technical systems. In this project we try to:

(1) investigate the nature and generation mechanisms of meaning in cognitive autonomous systems and also
(2) apply the emerging concepts in several research platforms with very different cognitive requirements and contexts: heterogeneity, scalability and visual awareness.

In this context, the project tries to build a formal theory of meaning to be applied in the definition of control mechanisms based on explicit representations of meaning. These mechanisms will be used in the design of self-aware control architectures for autonomous systems and will be implemented in the form of reusable software modules using standardized software deployment platforms.

Two domain targets are being used for the exlporation of meaning-centric cognitive architectures: the domain of process control systems (as described in (Sanz *et al.*, 2005)) and the domain of cognitive robotics.

In the context of this last one we are using a common mobile robot platform to build a cognitive control system atop of it. The platform runs common local software and is accessible to a collection of remote agents by means of CORBA technology. This base design enables the exploration of several architectural alternatives with a minimum of reengineering effort of the robotic platform (for example, we have integrated a SOAR-based system with the robot platform using the CORBA object wrappers).

### *4.2 Robot characteristics*

The robot selected is a well-known Pioneer 2AT8 robot (see Figure 3). The Pioneer 2AT-8 is a mobile robot made by ActivMedia Robotics. It is a sked-steering vehicle with lightweight aluminium body and four pneumatic wheels propulsed by four reversible DC motors, equipped with high-resolution optical encoders for precise position, speed sensing and advanced dead-reckoning. Electronic devices aboard are controlled by a Hitachi H8S microcontroller.

Fig. 3. Pioneer 2AT robot with wireless antenna.

The Pioneer 2AT robot has also two sonar arrays, each with eight transducers that provide object detection and range information for collision avoidance.

### *4.3 Wrapper design*

The Pioneer robot comes with a robotic sensing and control class library called ARIA. This library allows control of Pioneer robots from a computer connected through a serial link to the onboard Hitachi H8S microcontroller.

We have developed a CORBA wrapper to ARIA library. This initial implementation of the Pioneer CORBA wrapper (Pareja, 2004) provided remote access to the robotic platform by means of a collection of methods provided by a single CORBA object running on the robot computers.

CORBA broker technology, an IDL-specified interface and a servant implementation associated to this interface allow the creation of a server application that allows the remote control of the Pioneer robot. The IDL establishes a contract between clients and the Pioneer server that describes the types and object interfaces used by our application. The servant implementation contains calls to ARIA methods corresponding to remotely requested IDL operations. Mutexes are used when appropiate to avoid more than one thread, simultaneous control data modifications. Data structures defined in the IDL and used by some functions allow faster communication of multiple state, movement, position and sensing data in a single operation than multiple CORBA calls asking for data one by one.

### *4.4 RT wrapper implementation*

A new implementation of the wrapper has been developed for the SOUL project using Real-time

```
interface Pioneer2AT{
...
// Data exchanging methods --------------
Pioneer2AT_State getRobotState ();
Pioneer2AT_Movement getRobotMovement ();
Pioneer2AT_Position getRobotPosition ();
void getRobotSensing ( out Pioneer2AT_Sensing
sensing );
/ Robot fast state checking ------------
boolean isReady ();
boolean isConnected ();
boolean isRunning ();
boolean isStalled ();
boolean isEmergency ();
boolean isMoving ();
boolean isEnabled ();
boolean isBreaked ();
float getBattery ();
TimeStamp getTime();
void setTimeToNow();
...
```

Fig. 4. A portion of the IDL interface specification of the CORBA server Pioneer.

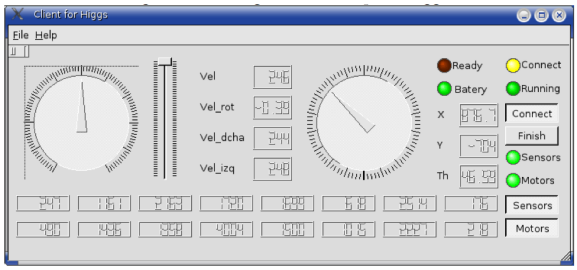


Fig. 5. A simple GUI client of the CORBA server Pioneer.

CORBA technology. This implementation will enable the implementation of coexisting independent applications interacting with the same robot platform. This new implementation leverages the RT CORBA control of resources to reduce the interference produced between applications.

The RT Pioneer server uses the following RT-CORBA features:

- Server declared priorities: to coordinate priorities across the distributed system.
- Threadpools with lanes: to reduce latencies in time-critical method invocation and eliminate race conditions and interference in the provision of thread resources for prioritised method invocations.
- Priority banded connections: to minimise interference in prioritised request management between real-time and non real-time invocations.

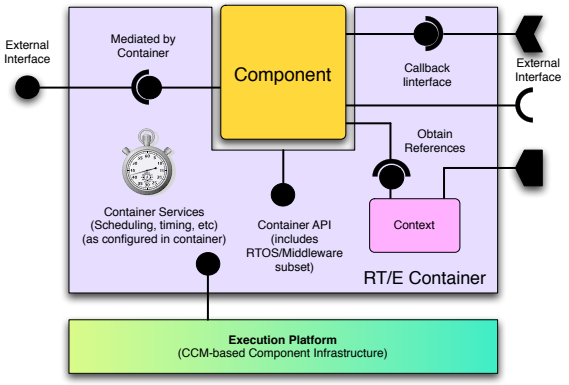


Fig. 6. Embbedable component technology improves encapsulation and functional/non-functional separation of concerns in real-time embedded systems while at the same time enables a simplified configuration and deployment of distributed applications.

## 5. CONCLUSIONS AND FUTURE WORK

CORBA technlogy has demonstrated its utility on the construction of distributed aplications in many domains. The Real-time CORBA specficiation —and, in particular, the RT CORBA brokers— have made possible the use of this technology in the implementation of networked controllers.

A CORBA object has been built in this line of work. This object wraps a Pioneer 2AT robot from Activmedia Robotics that enables the remote object interaction with parts of the ARIA class library. The RT CORBA implemantion of the broker incorporates resource control mechanisms to improve end-to-end predictability of the aplication.

Using these resources it is possible build a control system for the Pioneer robot where remote clients (e.g. the GUI client of Figure 5) do not interfere with networked control loops that run remotely to the robot. This reduced interference will make possible also the schedulability analysis of the aplication.

Ongoing work include the refactoring and reconstrucion of the system using embeddable component technology (see figure 6) based on the Lightweigh CORBA Component Model specification. More information in the IST COMPARE project website (www.ist-compare.org).

## REFERENCES


Hristu, Dimitrios and Levine, William S., Eds. (2005). *Handbook of Networked and Embedded Control Systems*. Birkhauser.

Huang, Hui-Min, Elena Messina and James Albus (2003). Autonomy level specification for intelligent autonomous vehicles: Interim progress report. In: *Proceedings of the 2003 Performance Metrics for Intelligent Systems (PerMIS) Workshop*. Gaithersburg, MD, USA.

OMG Robotics DSIG (2005). Robotic systems RFI. RFO mars/05-06-12.

Pareja, Iván (2004). Sistema de comunicaciones de la plataforma móvil pioneer 2at-8. Master's thesis. Universidad Politécnica de Madrid. Escuela Técnica Superior de Ingenieros Industriales.

Radermacher, Ansgar, S. Robert, C. Wigham, V. Seignole and R. Sanz (2005). State of the art in embedded commponent technology. IST COMPARE Project Deliverable 1.1.

Sanz, Ricardo, Fernando Matía and Eugenio A. Puente (1999*a*). The ICa approach to intelligent autonomous systems. In: *Advances in Autonomous Intelligent Systems* (Spyros Tzafestas, Ed.). Chap. 4, pp. 71–92. Microprocessor-Based and Intelligent Systems Engineering. Kluwer Academic Publishers. Dordretch, NL.

Sanz, Ricardo, Idoia Alarcón, Miguel J. Segarra, Angel de Antonio and José A. Clavijo (1999*b*). Progressive domain focalization in intelligent control systems. *Control Engineering Practice* **7**(5), 665–671.

Sanz, Ricardo, Jos Antonio Clavijo, Miguel Segarra, Angel de Antonio and Mariano Alonso (2001*a*). CORBA-based substation automation systems. In: *Proceedings of IEEE Conference on Control Applications*. Mexico D.F.

Sanz, Ricardo, Mariano Alonso, Ignacio López and Carlos A. García (2001*b*). Enhancing control architectures using CORBA. In: *2001 IEEE International Symposium on Intelligent Control*. Mexico D.F.

Sanz, Ricardo, Miguel Segarra, Angel de Antonio, Idoia Alarcn, Fernando Mata and Agustn Jimnez (2000). Plant-wide risk management using distributed objects. In: *IFAC SAFEPROCESS'2000*. Budapest, Hungary.

Sanz, Ricardo, Rafael Chinchilla, Manuel Rodriguez, David Perez and Carlos Martinez (2005). Pct: Component-based process control testbed. In: *44th IEEE Conference on Decision and Control and European Control Conference*. Seville, Spain.

Schmidt, Douglas C. and Fred Kuhns (2000). An overview of the real-time CORBA specification. *Computer* **33**(6), 56–63.

Segarra, Miguel J. (2005). CORBA Control Systems. PhD thesis. Universidad Politécnica de Madrid.