An Adaptive Model for Programming Distributed Real-Time Applications in CORBA

Carlos Montez[1] montez@lcmi.ufsc.br
Rômulo Silva de Oliveira[2] romulo@inf.ufrgs.br
Joni Fraga[1] fraga@lcmi.ufsc.br

[1]LCMI - Depto de Automação Sistemas - Univ. Fed. de Santa Catarina
Caixa Postal 476 - 88040-900 - Florianópolis - SC - Brazil

[2]Inst. de Informática - Univ. Fed. do Rio Grande do Sul
Caixa Postal 15064 - 91501-970 - Porto Alegre - RS - Brazil

Abstract

CORBA is a middleware with open standardization that is receiving plenty of acceptance for facilitating the distributed objects programming. CORBA is being extended through the specification of interfaces and necessary abstractions for supporting applications with real-time constraints. The use of CORBA in the contemporary real-time systems is adequate to meet the new requirements of flexibility and cost reduction in these systems. The new abstractions included in the CORBA specifications will enable a variety of programming models for real-time applications. This paper presents an adaptive programming model for distributed real-time applications using CORBA concepts.

Keywords*: Distributed systems, Real-time systems, CORBA, Real-time objects, Adaptive computing.*

1. Introduction

CORBA (Common Object Request Broker Architecture) is a middleware that objectives minimizing existing difficulties in distributed programming for heterogeneous environments. Its open specifications, standardized by OMG (Object Management Group), are receiving growing acceptance. The architecture of reference specified by the OMG [7] is composed by an ORB (Object Request Broker) and services and facilities objects. The ORB is responsible for managing transparently the communications among distributed objects. The services and facilities are standardized objects that support some basic functions used by the application objects.

Many distributed real-time systems are adopting the object orientation paradigm and using the CORBA concepts for building its programs. The final objective is to meet the new needs of portability, interoperability, flexibility and cost reduction of the contemporary real-time systems.

The use of CORBA, as well as other open architectures, in distributed real-time systems is a recent area of research. In spite of that, that tendency comprises a large range of applications, involving from systems that interacts with deterministic environments, such as embedded applications and control of processes; as well as large scale systems, characterized by a dynamic computational load, such as multimedia applications in the Internet. Many of those application domains require real-time guarantees (off-line or on-line) of the networks, operating systems and middleware components, in the sense of supporting its time constraints. However, in the current specifications, CORBA standards are inadequate to support real-time requirements.

In 1995, a Special Interest Group (SIG) has been formed within the OMG with the goal of extending the CORBA standard with support for real-time applications. The final specification, product of that standardization effort, will be adopted in the second semester of 1998. That final specification is intended to present several abstractions (interfaces and mechanisms) enabling the execution of real-time applications, allowing the definition of a variety of real-time programming models.

The study described in this paper is part of a research we are conveying with the objective of developing a programming model for distributed real-time applications using CORBA. The proposed model has adaptive characteristics in the sense that some scheduling decisions are based on the conditions of the system observed through monitoring. The adaptive technique of imprecise computation is implemented using multiple versions of methods. The model can be used in both deterministic or best-effort approaches. In deterministic environments, a minimum guarantee can be offered to the applications and the adaptation can be seen as a form of offering an additional quality in the services. In environments that does not offer predictability, in best-effort approaches, the adaptive characteristic of the model can be seen as a form of replying to the dynamic variation of the environment, obtaining several levels of quality in the offered services.

We are building the scheduling approach using the services and concepts that will be incorporate in the final CORBA specifications for real-time programming. The concepts of interceptors, threads management and explicit binding, for example, are used for monitoring invocations and to implement adaptive techniques.

This paper is divided in 6 sections. Section 2 describes some of the difficulties found for the execution of real-time applications using the current specifications of CORBA. It is also shown some of the main features that will be incorporate in the final specifications of real-time CORBA. Section 3 presents a discussion on adaptive models. Section 4 describes the proposed model. Section 5 lists some other researches related to ours model. Finally, section 6 presents the conclusions and indicates future directions of our works.

2. CORBA specification for real time

The key factor that makes CORBA not suitable for real-time execution is the fact that ORB, services and facilities objects, defined according to the current specifications, are not prepared for working with time constraints. CORBA was created targeting applications in general that desire transparency in distribution and in the way resources are managed. On the other hand, those transparencies in real-time applications, are not desirable neither necessary. For example, location and migration transparencies of objects usually are not appropriate for real-time applications.

The interoperability problem involving different ORBs is solved in CORBA by specifying an interoperability protocol for Internet — the IIOP (Internet Inter-Orb Protocol). That protocol is designed for general purpose systems, sending messages in standardized formats using TCP/IP connections. The lack of predictability features in TCP makes the IIOP an inadequate protocol for several real-time applications.

Also, CORBA doesn't have some characteristics for real-time programming, such as invocation with timeout. Besides, the oneway invocation in the current standard with asynchronous semantic, is insufficient for the needs found in real-time programming.

The lack of middleware support for real-time applications, lead the OMG in 1995 to create a work group with the objective of extending the CORBA standard (the RT-CORBA). In September of 1997 a RFP (Request for Proposal) document [6] was published requesting proposals of standardization. In January of 1998, five proposals [1-5], result of the two year-old efforts, were presented. Those proposals will compose one standard that will be consolidated until the end of 1998.

The proposals emphasize the definition of ORB mechanisms instead of policies. The specification of a select group of policies probably would not satisfy all the real-time applications. On the other hand, the definition of mechanisms allows the composition of specific policies for each application. All five proposals, in spite of their differences in interfaces and in the use of some concepts, offer several mechanisms and abstractions in common, that were suggested originally in the RFP document. Some of those mechanisms are used in our programming model.

The main mechanism proposed for ORB with the objective of enabling the construction of adaptive execution environments to the specific needs of an application is the interceptor. Interceptors are application objects that have its operations called by the ORB in different instants of an invocation. An interceptor in the client side is activated when an invocation is made and when the answer is received from the server. An interceptor in the server side, is activated when an invocation is received and when the answer is sent. The interceptor was initially specified in the CORBA security service, and recently was standardized by OMG in the CORBA 2.2 [7]. The proposals for RT-CORBA (real-time CORBA) extends the interceptor interfaces, creating other interceptors categories, in the object adapter, the transport layer and in the threads management.

Threads, pool of threads and request queues are some resources that applications can reserve and manipulate directly. For example, a pool of threads can be created to execute the requests of methods received by a server. A request queue can be created and associated with that threads pool. The ordering in those queues can be specified as FIFO, based on global priorities, or any other policies specified in the application and implemented through the interceptors mechanism.

The notion of explicit binding, presented in the proposals, relates to the establishment of an association (a communication path) between clients and servers objects. During the binding, several steps must be made to interconnect the objects, such as, to locate target objects and to initialize structures of data that support the communication among objects. On the client side, the binding can be used to guarantee that its timeliness requirements are respected. For example, through the selection of an appropriate transport protocol and of a timeout value for fails detection. On the server side, the binding allows that the necessary resources, such as threads and queues, for the execution of the requests, are allocated.

3. Adaptive models

The need of strict predictability in hard real-time systems, where all the deadlines must be satisfied, implies in limited programming models. In this case,

almost the decisions are generally made off-line. Those models assume that the worst case of using resources (CPU, disk access, messages delays, etc.) are defined and known before execution. For example, the determination of the worst case execution time (WCET) of tasks is an arduous work that demands a priori knowledge of the execution environment. It can be only determined analytically by means of restricting the use of certain dynamic constructions of programming languages (recursive call, for example). Using simulations for determining the WCET does not guarantee that the worst cases were really obtained.

The reservation of resources necessary in hard real-time systems leads to a great sub-utilization of resources. That is a "price to be paid" for the guarantee obtained by those systems. On the other hand, in soft real-time systems, tasks can receive a best-effort approach to meet its time constraints. A simple example of best-effort approach is the use of average time of computation in the timeliness analysis of the system, accompanied by the discard of tasks when necessary. That approach results in a better use of resources, under the restriction it eventually loses deadlines.

There are systems with hard real-time constraints that operate for long periods in non deterministic environments. Military, radar and air traffic control exemplify some of those systems. They should possess both a previsible and a dynamic and flexible behavior, using techniques that allow adaptations to the changes imposed by the non deterministic environment.

The imprecise computation [18] is one technique found in the literature that allows the combination of deterministic guarantee with graceful degradation in the occurrence of overload. That technique separates computation in a mandatory and an optional part. The mandatory part, off-line guaranteed, is capable of generating a result with a minimum quality, while the processing of the optional part refines this result. During an overload, a "minimum" level system operation can be guaranteed in a deterministic way by executing only the mandatory parts.

The scheduling approaches where decisions are taken dynamically, according to results of the environment monitoring, as is usually done in the imprecise computation technique, are denominated adaptive scheduling. A series of adaptive approaches are being proposed with the same objective of the imprecise computation: guaranteed services and graceful degradation [8-12]. Besides the already mentioned difficulty in the obtaining of WCET, some other reasons can be listed for the use of adaptive approaches:

i) Tasks with real-time constraints define a range of values for its periods and deadlines where it operates in a satisfactory way. Furthermore, in a distributed real-time system, tasks impose time constraints values in end-to-end basis. It is an engineering work to attribute intermediary values [20]. Conventional approaches of real-time scheduling usually fastens some values as if these were the only ones possible for the correct operation of the application.
ii) Some real-time periodic tasks tolerate the eventual losses of some deadlines if those losses are spaced [11]. For example, in anti-locking brake systems, a real-time task typically determines the beginning of a lock, repetitively monitoring the speed of each wheel. In the case of missing a periodic activation deadline, it is possible to determine the speed of a wheel, projecting the obtained values of the previous activations of the task.

The existent approaches for soft real-time tasks, such as those used in multimedia, generally are not appropriate to be used in other types of real-time tasks because the predictability obtained is very precarious. The main objective of the new adaptive approaches is to obtain deterministic predictability in normal conditions of operation, and a controlled service degradation during an overload.

Those adaptive approaches bring some additional benefits, because its use make possible the supply of dynamic guarantees. For example, it may be assumed that tasks can arrive dynamically and need guarantees before beginning to execute. In the case of not being possible to accept those tasks, instead of rejecting them (as it happens in conventional admission tests), the system can accept them by degrading other services of some previous accepted tasks. Such approach is proposed in [8].

The technique of imprecise computation considers a model of tasks where the obtained benefits increase when tasks receive larger times of computation. The new approaches of adaptive scheduling, in general, enlarges that concept also considering models of tasks where the benefits obtained increase when tasks are activated more frequently. That characteristic comes from the direct observation that control application tasks have freedom to operate in a range of frequencies. For example, in tracking applications, the more close it is the vehicle tracked by the radar, bigger should be the frequency of the tasks that maintain the data [11]. Multimedia applications also use that idea. In audio transmissions it is possible to improve the obtained quality by increasing the sampling rate.

Adaptive techniques as the imprecise computation and the task frequency manipulation try "to combat" the cause (overload) and not "to minimize" its symptoms in the occurrences of temporal failures. Reducing the occurrence of overloads also reduces its consequences: deadlines losses, discard of messages in the communication support, discard of periodic activations of tasks, and (complete) discard of tasks.

4. An adaptive programming model for RT-CORBA

The final specification of real-time extensions for CORBA will present several abstractions (interfaces and mechanisms) enabling the execution of real-time applications. Those abstractions will allow the building of a variety of real-time programming models — from deterministic to best-effort models. This text presents the APMC (Adaptive Program Model for CORBA) — a adaptive model based on real-time objects to be used in CORBA.

4.1 Real-time objects

In spite of CORBA being a distributed object architecture, it does not demand the use of object oriented models and programming languages. However, the object paradigm, used in many areas of computing, also presents several attractiveness for distributed programming, making natural the development of several models of distributed objects. In the context of distributed real-time systems, the object paradigm also presents several advantages, facilitating the understanding and management of the complexity of those systems, leading to the recent development of several distributed real-time objects models [13-15].

The APMC is an object oriented model that supplies an integrated vision, where methods and abstractions of conventional objects can be used also by the real-time objects, and vice-versa. That characteristic allows code reutilization, and increasing of the productivity. It allows real-time applications to use some conventional services such the CORBA name service and trading. However, it is important to note that in the requests of methods involving conventional objects, it is not possible to specify and enforce time constraints, even if the requests are done by real-time objects.

4.2 Meta-object protocols

APMC offers a clear separation between the functional and the timeliness aspects in the implementation of a real-time object, allowing real-time objects to be reused easily with other timeliness constraints. That is possible through rewriting the code that implements the timeliness aspects. Besides, real-time objects can be reused in the form of conventional objects, discarding the part of the code that treats the timeliness aspects. It is also possible to reuse conventional objects. They can be wrapped in real-time objects, by writing the code about the timeliness aspects.

The separation between concerns of functional aspects (structural) from non-functional aspects (behavior), allows a better code reutilization, and facilitates the development of complex systems. The functional properties of the application can be programmed by people that understand the application, while the programming regarding real-time scheduling, fault tolerance, synchronization, and so on, can be driven by programmers specialist in those areas [21].

The separation of functional from timeliness aspects is made using the computational reflection paradigm, and is implemented using meta-object protocols. A reflective system is a system that incorporates structures that represent and implement aspects of itself. The meta-object model introduced in [19] explicitly separates objects in different levels: base and meta (Figure 1). For each real-time object, a meta-level object exists (meta-object) that implements the behavior aspects (time constraints) of the object in the base level (base object).

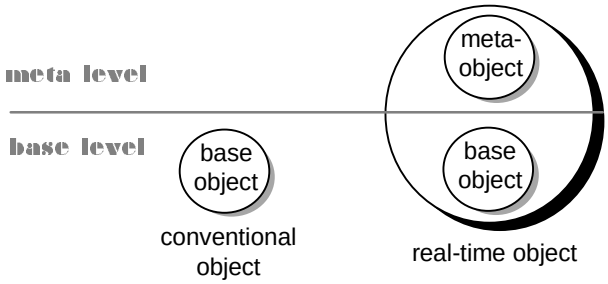

Figure 1. Conventional and real-time objects in the model APMC.

The meta-object is activated when receiving a request of a method driven to its respective base object. The request is intercepted and deviated to the meta-object that may execute some computations in the meta-level objectifying to meet the timeliness of the activated method (Figure 2). Later on, the meta-object can send the request for the method of the base object activated originally. APMC supports both synchronous (request-reply) or asynchronous (request-only) interactions among objects. In the case of the synchronous interaction, the reply of the base object server can also be intercepted and deviated for the meta-level, before it returns to the client that made the request (only in the case of the computation made in the meta-level not deciding to raise an exception, instead of returning the reply to the client). When necessary, on the client side, the meta-object can be activated in the moment that the request is done, and also when the reply (in the case of synchronous interaction) or exception is received.

The requests interception allows pre-processing and pos-processing for the meta-level in the sense of controlling the timeliness requirements (or other behavioral aspects) specified by the programmer. However, some applications may be interested on the interception of other events during the processing of base object methods. For example, if computational reflection is used with the objective of monitoring the behavior of a certain application, the interception of requests is not enough because it represents a big granularity in the

monitoring. Ideally, one should monitor each state change of the threads that execute the methods of the base objects. The model APMC allows a meta-object to be activated at each change in the state of the method (thread) of the corresponding base object. That feature is optional for representing an overhead that must be accounted in the computation times of involved methods in base object.

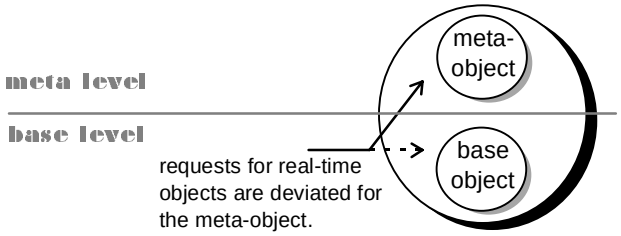

Figure 2. Requests for a real-time object.

The new interceptor categories, that will be incorporated in CORBA, are important for the implementation of meta-objects in the APMC model. Server interceptors are used for catching requests to the methods of the base objects and to deviate for the meta-object. Thread interceptors are responsible for monitoring changes of state of the threads that execute the methods in the base objects.

4.3 Active and passive methods

Objects in the APMC model possess methods that can be active or passive. Passive methods are only executed when receiving requests (synchronous or asynchronous) of other methods. The active methods already possess a thread of execution, and they are executed since the moment the object is created.

An object can have internal concurrence in its methods through the declaration of more than one active method. However, the race conditions control in the access of shared resources inside the object, are a subject that shall be treated by the application code and it doesn't constitute a concern of the model APMC. The conventional concept of active object, can be implemented in the model APMC through the declaration of one active method in each object.

Real-time objects can implement time-triggered tasks (periodic activations of an active method) in the meta-level. Passive methods may be used to implement event-triggered tasks, that are activated through external events (arrival of requests). Time-triggered and event-triggered methods can model several types of interactions found in real-time applications. For example, in control applications some tasks usually are responsible for monitoring periodically values of sensors and resending the results to other tasks that are responsible for activating tasks that control certain devices. Those applications can be modeled considering the methods that monitor the sensor as active time-triggered methods and the others as passive event-triggered methods (Figure 3).

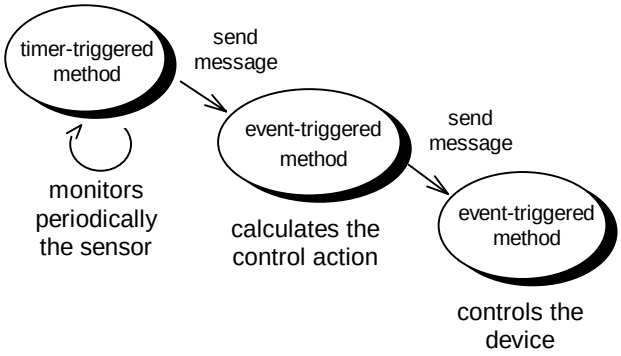

Figure 3. Control application modeled using active and passive methods.

Several other kinds of interactions among distributed tasks, such as the traditional client/server and producer/consumer, can also be modeled by using active and passive methods.

The new standardized interface for managing threads in the real-time ORB specifications, will allow a larger portability in the implementation of active and passive methods. The treatment of the threads will be done in an uniform way, even if the execution platform has Solaris, NT, or POSIX.1c threads. Each declared active method, will have a thread allocated for itself in the moment that the object is created. Each passive method will (i) be executed in the thread of the active method that originated the request, or (ii) have its thread created dynamically at the moment of the arrival of the request, or (iii) have its thread removed from a pool previously allocated.

The option (i) will happen whenever an active method makes request to a passive method in the same object. The options (ii) or (iii) represent alternatives in the form of activating a passive method when the request is originated from another object. The option (iii) is faster and with more predictability (when the number of existent threads in the pool is enough to treat all the simultaneous requests for the passive methods of the object).

4.4 Adaptation in the APMC

The technique of imprecise computation through multiple versions can be easily implemented in the model APMC. In an implementation form (denominated in [13] as time polymorphic invocation), several methods are implemented in the base object offering the same service with different quality levels (QoS) and, consequently, with different processing times associated. When intercepting a request, the meta-object of the server can select alternative methods in the base object depending on the conditions of the moment (for

example, a client deadline for the request).

Figure 4 represents an example of a control application implemented using the multiple versions technique. The request-only (without reply) sent by the active time-trigger method that monitors the sensor is reflected for the method in the meta-object that selects the base method implementation dynamically, according to the current conditions.

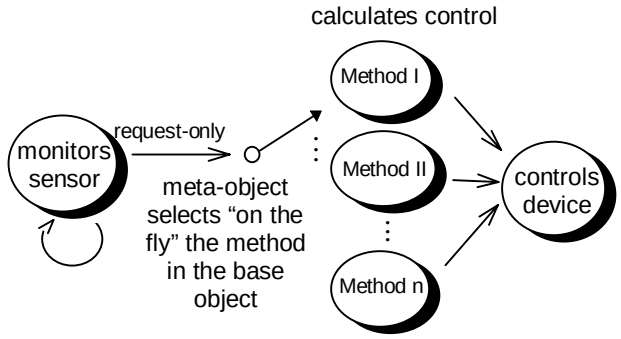

Figure 4. Control application implemented using multiple versions.

In environments that offer predictability, "Method I" could represent a method that returns a minimum acceptable quality (guaranteed off-line by schedulability analyses), while the other methods could represent better quality levels for the control. In environments without predictability, each method could represent a growing level of quality, however the application will not receive any guarantee of a "minimum operation".

The concepts of interceptors, pool of threads and request queue that will be incorporated to the real-time ORB are important for the implementation of adaptive mechanisms in APMC. A pool of threads is created in an object server and associated to a request queue. A server interceptor is responsible for receiving and deviating the requests for the meta-object. The requests deviated for the meta-object are queued in the request queue. The server interceptor sort (schedule) the queue in accordance to a specific policy. The threads catch the requests of the top of the queue and execute them in behalf of the client.

The global priority concept [17] can influence the policies of ordering the request queues and of the dynamic attribution of priorities to the threads. For example, a client can dynamically specify time constraints for the invocation, just as a deadline. Those time constraints are used for calculating the global priority value of the request. That global priority value is used for scheduling the method (thread) executed in the server.

4.5 Predictability in the APMC

Due the existence of large and complex systems and the heterogeneity of theirs components, the use of distributed real-time systems in CORBA may be confused with one presenting dynamic load and executing on a hostile environment. However, CORBA can also be applied in static systems, characterized by a priori knowledge of the computational load and of the available resources for execution. In those conditions the execution of time critical applications is possible.

To obtain predictability in a distributed system is necessary a notion of global time and a maximum delay for communication among objects. New technologies, like GPS receivers (for obtaining a global time based on UTC), ATM networks (that allow the specification of QoS parameters) and POSIX.1b and POSIX.1c specifications (that standardize several real-time mechanisms in the operating system level) enable to obtain predictability in distributed systems. APMC can be used with those technologies in the sense of obtaining predictability in the execution of a real-time system.

New CORBA abstractions, incorporated in the specifications of real-time ORB, are also valuable for obtaining deterministic predictability. For example, the explicit binding concept allows objects to determine a priori the communication maximum delays. That aspect is fundamental to allow an off-line schedulability analysis of the system.

5. Related works

The DHDA project [17] was the pioneer work in the sense of establishing the needs of real-time in CORBA. It influenced plenty the work of OMG. It specifies a best-effort approach to support end-to-end time constraints in a dynamic environment, with objects being added and removed, and with changing timing constraints.

TAO [16] is a real-time ORB that, in deterministic environments with fixed load, can offer deterministic guarantee enabling the execution of hard real-time applications. The CORBA IDL is extended to allow the description of timing constraints associated with each method in interface declarations. A real-time scheduling service is implemented using the rate monotonic policy, considering only independent threads.

RTRD [14] uses meta-object protocols on CORBA, allowing applications to control its functionalities based on its timeliness requirements. That quite flexible model can work with several types of timing constraints that are specified by the applications. The best-effort approach considers only soft real-time tasks.

In [12], CORBA interceptors are used to monitor requests to CORBA objects, and the result of that monitoring feedbacks an adaptive scheduling approach (task-pair scheduling). The task-pair scheduling is an implementation of the imprecise computation with two versions. During the execution of one of the versions, the

monitoring results can lead to the decision of canceling the execution and executing other version with a small code that represents an exception treatment.

The QuO project [10] establishes a programming model that extends the IDL CORBA to allow the specification of routines for treatment of changes in the execution environment conditions. The objective is the implementation of very adaptive applications that can execute totally in dynamic environments as the Internet.

The model APMC incorporates several characteristics of those works. Differently of [10], [14], [16] and [17], the APMC embraces both applications that need best-effort guarantees with adaptive characteristic, and applications that need deterministic guarantees. The model in [12] is similar to APMC, except that it does not use the new real-time ORB mechanisms. An important characteristic of the APMC is that it does not present any proposal of extension to the CORBA standard. The APMC is totally integrated with the new real-time CORBA specifications that are being standardized by OMG.

6. Conclusions

With the appearance of new abstractions offered by the real-time ORB that is being specified by OMG, several kinds of models for real-time applications programming can be developed. This paper presented the APMC — an adaptive object oriented model for real-time programming in CORBA. The APMC allows the implementation of adaptive approaches through the incorporation of the time polymorphic invocation technique. That technique can be used both in deterministic and best-effort approaches. The separation between the implementation of functional aspects and the real-time constraints is made using computational reflection, defining explicitly separated levels: base and meta. The programmer can implement the specific timeliness requirements of its application in the meta-object of each real-time object.

Now, the main effort in our project is to adapt a complete scheduling framework to our model. We are interested that the framework allows the programmer to specify execution patterns in the form of imprecise computation and (m,k)-firm guarantees [9]. We are also studying, as a secondary objective, a way to incorporate, other adaptive techniques, like variations in the activation frequency of time-triggered methods.

7. References

[1] Alcatel, HP, Lucent, OOC, Sun, Tri-Pacific, “Realtime CORBA - Version 1.0”, Object Management Group (OMG), Document orbos/98-01-08, Jan. 1998.

[2] Northern Telecom, Iona Technologies, “Realtime CORBA Extensions”, Object Management Group (OMG), Document orbos/98-01-09, Jan. 1998.

[3] Highlander Communications, Visigenic Software, “Realtime CORBA”, Object Management Group (OMG), Jan. 1998.

[4] Lockheed Martin Federal Systems, “Realtime CORBA, Response to OMG RFP for Realtime CORBA Extensions”, Object Management Group (OMG), Document orbos/98-01-04, Jan. 1998.

[5] OIS, “Realtime CORBA, Initial Submission”, Object Management Group (OMG), Jan. 1998.

[6] OMG Realtime Platform SIG, “Realtime CORBA 1.0 RFP”, Object Management Group (OMG), Document realtime/97-09-31, Sep. 1997.

[7] OMG, “The Common Object Request Broker: Architecture and Specification - Revision 2.2”, Object Management Group (OMG), Feb. 1998.

[8] T. Abselzaher, E. Atkins, K. Shin, “QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control”, *Proc. of IEEE RTAS´97*, Montreal, Canada, Jun. 1997.

[9] P. Ramanathan, “Graceful Degradation in real-time control applications using (m, k)-firm guarantee”, *Proc. of Fault-Tolerant Computing Symposium*, 1997, pp. 132-141.

[10] J. A. Zinky, D. E. Bakken, R. D. Schantz, “Architectural Support for Quality of Service for CORBA Objects”, *Theory and Practice of Object System*, Vol. 3(1). 1997.

[11] T. Kuo, A. K. Mok, “Incremental Reconfiguration and Load Adjustment in Adaptive Real-Time Systems”, *IEEE Trans. on Computers*, Vol. 46, No. 12, Dec. 1997.

[12] M. Gergeleit, E. Nett, M. Mock, “Supporting Adaptive Real-Time Behavior in CORBA”, *Proc. of 1st IEEE WMDRTSS*, San Francisco, CA, Dec. 1997.

[13] K. Takashio, M. Tokoro, “DROL: An Object-Oriented Programming Language for Distributed Real-Time Systems”, *Proc. of OOPSLA’92*, 1992, pp. 276-294.

[14] O. Furtado, F. Siqueira, J. Fraga, J-M Farines, “A Reflective Model for Real-Time Applications in Open Distributed Systems”, *Proc. IFIP/IFAC WRTP´96*, Brazil, Nov. 1996.

[15] G. Li, D. Otway, “An Open Architecture for Real-Time Processing”, APM, Poseidon House, Castle Park, Cambridge, Document, APM.1270.02, Oct. 1994.

[16] D. C. Schmidt et al., “TAO: A High Performance Endsystem Architecture for Real-Time CORBA”, *IEEE Communications Magazine*, 14(2), Feb. 1997.

[17] V. F. Wolfe, et al., “Expressing and Enforcing Timing Constraints in a Dynamic Real-Time CORBA System”, University of Rhode Island, Department of Computer Science and Statistics, Technical report TR97-252, Jun.1997.

[18] J. W. S. Liu, W. -K Shih, K. -J. Lin, R. Bettati, J. -Y. Chung, “Imprecise Computations”, *Proceedings of the IEEE*, Vol. 82, No. 1, Jan 1994, pp. 83-94.

[19] P. Maes, “Concepts and Experiments in Computational Reflection”, *Proc. of OOPSLA’87*, 1987.

[20] R. Gerber, S. Hong, “Guaranteeing Real-Time Requirements with Resource-Based Calibration of Periodic Processes”, *IEEE Trans. On Software Engineering*, 21(7), Jul. 1995.

[21] P. M. Melliar-Smith, L. E. Moser, P. Narasimhan, “Separation of Concerns: Functionality vs. Quality of Service”, *Proc. 3.o Workshop on Object Oriented Real-Time Dependable Systems*, Ca, Feb. 1997.