Extensões dos Padrões CORBA para Aplicações de Tempo Real

Carlos Montez Joni Fraga Jean-Marie Farines
LCMI - Depto de Automação e Sistemas
Univ. Fed. de Santa Catarina
Caixa Postal 476
CEP 88040-900 - Florianópolis - SC
{montez, fraga, farines} @lcmi.ufsc.br

Rômulo Oliveira
Inst. de Informática
Univ. Fed. do Rio Grande do Sul
Caixa Postal 15064
CEP 91501-970 - Porto Alegre - RS
romulo@inf.ufrgs.br

Resumo

O CORBA é um *middleware* com padronização aberta que vem recebendo bastante aceitação por facilitar a programação na forma de objetos distribuídos. Entretanto, o CORBA não oferece a previsibilidade necessária para o desenvolvimento de aplicações de tempo real. Devido a isso, algumas propostas estão sendo feitas no sentido de especificar interfaces e abstrações necessárias para tempo real, e até o final deste ano, a especificação final estará completada. Este artigo discute as várias direções no encaminhamento das extensões CORBA para tempo real.

Abstract

CORBA is a middleware with open standardization that is receiving plenty of acceptance by facilitating the programming in the form of distributed objects. However, CORBA doesn't offer the necessary previsibility for the development of real-time applications. Some proposals are being made with objective of specifying interfaces and necessary abstractions for real-time processing, and the final specification will be completed until the end of this year. This article discusses the several directions in the future adoption of the CORBA extensions for real time.

1. Introdução

Sistemas abertos abrangem um significante leque de tecnologias e especificações que, de forma diversa aos ambientes proprietários, permitem aos usuários escolherem de um vasto grupo de tecnologias de *hardware* e *software*, quais se adaptam melhor às suas necessidades. Nos últimos anos sistemas distribuídos têm experimentado grandes avanços. O enfoque atual é a utilização de padrões como CORBA, ATM, POSIX e componentes de prateleira (*off the shelf*), em detrimento de soluções e protocolos proprietários.

O CORBA (*Commom Object Request Broker Architecture*) é um *middleware* criado com objetivo de permitir que as dificuldades existentes na programação distribuída em ambientes heterogêneos sejam superadas. Suas especificações abertas, padronizadas pela OMG (*Object Management Group*) [OMG 95], vêm recebendo crescente aceitação. A arquitetura CORBA é formada por um ORB (*Object Request Broker*) e por objetos de serviço e de facilidades. O ORB é responsável por gerenciar de forma transparente as comunicações entre objetos distribuídos. A OMG também padroniza um conjunto de objetos de serviços e de facilidades comuns que suportam algumas funções básicas usadas por objetos da aplicação.

Muitas aplicações de tempo real estão abraçando o paradigma de orientação a objetos e considerando a utilização de CORBA como uma forma de se adequar a sistemas abertos. A utilização de CORBA, assim como outros padrões abertos, vem exatamente cobrir as novas necessidades de redução de custos, portabilidade, interoperabilidade e flexibilidade dos sistemas de tempo real contemporâneos.

A utilização de arquiteturas abertas em sistemas distribuídos de tempo real é uma área de pesquisa recente. Essa tendência é abrangente envolvendo desde sistemas que interagem com ambientes deterministas, tais como os formados por aplicações embarcadas e controle de processos, até sistemas de larga escala, caracterizados por uma carga computacional dinâmica, tais como aplicações de multimídia na *Internet*.

Muitos desses domínios de aplicação citados acima, requerem garantias de tempo real (dinâmicas ou em tempo de projeto) das redes de comunicação dos sistemas operacionais e de componentes de *middleware* no sentido de atender suas restrições temporais. Entretanto, nas especificações atuais, os padrões CORBA têm se mostrado inadequados para suportar

requisitos de tempo real.

Por esse motivo, desde 1995 a OMG estuda formas de estender as especificações CORBA no sentido de se obter um padrão CORBA para tempo real. Esse esforço de padronização vem seguindo um cronograma previamente estabelecido, e uma especificação deverá ser adotada até o final deste ano.

O objetivo desse artigo é discutir as várias direções no encaminhamento das extensões de CORBA para tempo real. Na seção 2 desse texto são levantadas deficiências nas especificações atuais do CORBA para atender requisitos de tempo real. São apresentadas na seção 3 experiências prévias de extensão do CORBA cujas idéias têm influenciado as propostas de padronização nos grupos de discussão da OMG. A seção 4 é dedicada à descrição das cinco propostas recentemente apresentadas à OMG como candidatas a padrão para tempo real. Essas propostas deverão compor uma única proposição que terá sua consolidação como padrão no final desse processo. Finalmente, a seção 5 desse artigo faz uma discussão sucinta dessas propostas, e os trabalhos de pesquisas referentes à adequação do *middleware* CORBA para aplicações de tempo real. No final, são indicados os trabalhos desenvolvidos no LCMI relacionados com o CORBA RT.

2. CORBA e aplicações de tempo real

A falta de suporte de *middleware* para aplicações de tempo real levou a OMG a criar um calendário no sentido de estender o padrão CORBA. Em 1996, dois textos foram publicados [OMG 96, OMG 96b] com objetivos complementares de levantar as capacidades desejadas em CORBA para atender requisitos de tempo real. Nesta seção, inicialmente, são examinadas algumas deficiências encontradas nas especificações atuais para o uso em aplicações de tempo real. Essas deficiências são extraídas de diversos textos [Schmidt 97, Wolfe 97, Montez 97, Yang 98]. Na seqüência são apresentados os principais requisitos definidos pela OMG no sentido da definição de um CORBA RT.

2.1. Deficiências do padrão CORBA atual

O CORBA foi criado para aplicações de propósito geral que desejam obter transparências de distribuição e na forma de gestão de seus recursos. Por outro lado, esses requisitos de transparências de distribuição em aplicações de tempo real, ou não são desejáveis ou não são necessários. Transparências de localização e migração de objetos, por exemplo, geralmente não são adequadas em tempo real. Os ORBs, serviços e facilidades CORBA, definidos segundo as especificações atuais, não estão também preparados para lidar com restrições temporais.

O problema de interoperabilidade entre diferentes ORBs é resolvido em CORBA, através da especificação de um protocolo de interoperabilidade IIOP (*Internet Inter-Orb Protocol*) padrão para *Internet*. Esse protocolo é projetado para sistemas de propósito geral, utilizando mensagens em formatos padronizados, trocadas usando conexões TCP/IP. A falta de previsibilidade no TCP torna esse protocolo inadequado para muitas aplicações de tempo real.

O CORBA não possui algumas características importantes para programação tempo real, tais como a possibilidade de invocação assíncrona e invocação com controle de *timeout*. A invocação *oneway* existente no padrão atual que possui semântica assíncrona, é insuficiente para as necessidades encontradas na programação tempo real.

2.2. Requisitos CORBA para tempo real

A OMG através do documento RFI (*Request For Information*) [OMG 96] definiu um conjunto de requisitos que devem ser mantidos por um ORB que ofereça suporte a aplicações com restrições de tempo real. Esses requisitos, que formam a base para a definição de um CORBA RT, são subdivididos em três grupos distintos: para o ambiente operacional, para a arquitetura ORB e para os serviços CORBA.

Requisitos CORBA para o ambiente operacional

- *Sincronização de relógios e atraso de comunicação delimitado* - Para se conseguir o

cumprimento de restrições temporais em uma base fim a fim, é necessária uma noção comum de tempo, implicando na necessidade de sincronização de relógios no sistema distribuído. Pelo mesmo motivo, é necessário o conhecimento dos atrasos máximos das mensagens trocadas entre objetos. Nesse último caso, os mecanismos de comunicação subjacentes devem assegurar limites máximos para os atrasos ou ainda, alguma forma de estatística referente a esses atrasos.

• *Escalonamento baseado em prioridades e herança de prioridades* - A maioria das abordagens de escalonamento de tempo real são baseadas na atribuição de prioridades. Dessa forma, o ambiente operacional subjacente ao CORBA deve suportar escalonamentos de tarefas (preferencialmente preemptivo) e gestões de filas baseados em prioridades. Protocolos de herança de prioridade [Sha 90] são requeridos para delimitar o tempo máximo de bloqueio das tarefas nessas abordagens baseadas em prioridades.

• *Monitoramento de recursos* - Esse requisito é importante, principalmente em abordagens que oferecem garantia dinâmica (*best-effort*), onde um ORB determina, baseado em estatísticas, se um recurso pode atender às restrições temporais de uma dada requisição. No monitoramento, os eventos que devem ser contabilizados incluem eventos internos ao ORB, tais como invocações de métodos e criações de *threads*.

As especificações para o ambiente operacional determinam ainda a necessidade da concorrência ao nível de *threads* e um suporte de comunicação, com vários protocolos, permitindo a definição de seus serviços com base em parâmetros de QoS.

A maioria desses requisitos apresentados nesse documento inicial, são cobertos quando o sistema operacional subjacente é compatível com as normas POSIX para tempo real.

Requisitos para o ORB e serviços CORBA

• *Transmissão de restrições temporais nas invocações de métodos* - Os ORBs devem permitir aos clientes especificarem restrições de tempo real nas invocações de métodos. Essas informações são transferidas junto com a própria invocação, podendo ser utilizadas pelos ORBs, sistemas subjacentes, adaptadores de objetos e servidores, para fazer cumprir os requisitos temporais.

• *Exceções específicas para tempo real* - Os mecanismos de exceção CORBA devem ser estendidos para indicar ocorrências de exceção relacionadas com restrições temporais.

• *Suporte para polimorfismo de desempenho*[1] - Esse requisito especifica que a seleção de uma implementação de um método, correspondendo a uma invocação específica, seja determinada dinamicamente com objetivo de atender as restrições temporais estabelecidas.

• *Serviços para atribuição de prioridades globais* - É desejável a criação de um serviço que estabeleça valores de prioridades "globais" coerentes através de todo o domínio do ORB. Permitindo, dessa forma, que os escalonadores possam estabelecer ordens nas execuções das invocações quando as mesmas competem por recursos no sistema distribuído.

Nas definições do RFI do CORBA RT, a OMG especifica requisitos de tempo real também para o *serviço de eventos* e *serviço de controle de concorrência* do CORBA [OMG 97b]. As especificações atuais do CORBA já definem um *serviço de tempo* [OMG 97b], entretanto suas interfaces precisam ser estendidas com métodos que permitam às aplicações obterem informações da "qualidade" do tempo fornecido pelo serviço subjacente de sincronização de relógios (por exemplo, se o resultado da sincronização é determinista ou estatístico).

3. Experiências prévias de extensão do CORBA

Atualmente, existem alguns ORBs comerciais portados sobre uma série de sistemas

[1] O polimorfismo de desempenho é uma técnica descrita em [Takashio 92] que se baseia na existência de diferentes implementações (com diferentes "qualidades" nos resultados retornados) para um mesmo método.

operacionais de tempo real. Entretanto, suas implementações não diferem das convencionais. As deficiências das especificações atuais do CORBA, levaram a alguns grupos de pesquisa desenvolverem propostas de extensões para tempo real. Duas dessas propostas, descritas em [Wolfe 97, Schmidt 97], influenciaram os debates da OMG e suas soluções, em sua maior parte, fazem parte de uma proposta de padronização apresentada em [Alcatel 98].

3.1. Projeto de Rhode Island

A Universidade Rhode Island em colaboração com a MITRE Corporation vêm desde 1994 pesquisando extensões do CORBA para tempo real [Wolfe 97]. Esse projeto especifica uma abordagem para suportar restrições temporais fim a fim em um ambiente dinâmico, onde clientes e servidores podem ser adicionados e removidos, e suas restrições temporais podem mudar com o tempo. Além do *serviço de tempo* CORBA, implementado para assegurar que as informações de restrições temporais se mantenham coerentes entre nós, esse projeto implementa o *serviço de eventos de tempo real*, o *serviço de controle de concorrência de tempo real*, e o *serviço de prioridades globais* citados nos requisitos da seção 2.2.

Devido a esse ambiente dinâmico, onde análises temporais em tempo de projeto são inviáveis, uma abordagem de melhor esforço (*best-effort*) é implementada, onde o objetivo principal é reduzir a porcentagem de *deadlines* perdidos. A política de escalonamento implementada é uma variação do EDF. Durante as invocações, restrições temporais (*deadlines*) e informações de importância são usadas no cálculo de uma prioridade global no nó cliente.

Quando uma requisição alcança um servidor, a mesma é executada por uma *thread*, criada com uma prioridade no sistema operacional do servidor que reflete a prioridade global do método invocado. O mapeamento da prioridade global na faixa de prioridades de um sistema operacional, pode provocar a existência de mais de uma *thread* com a mesma prioridade. Para reduzir possíveis problemas de inversão de prioridade, nos casos onde a faixa de prioridade global é muito maior que a faixa de prioridades do sistema operacional, os autores propõem heurísticas que aprimoram esse mapeamento.

As pesquisas estão sendo desenvolvidas usando sistemas operacionais em conformidade com POSIX, conectados por redes ATM. Para obter uma determinada previsibilidade no suporte de comunicação, o projeto enfatiza o uso do *serviço de latência de rede*, implementado sobre redes ATM, como forma de permitir a clientes a obtenção de estimativas e/ou garantias nos atrasos máximos das mensagens.

3.2. TAO

TAO é um ORB de alto desempenho para tempo real que está sendo desenvolvido na Universidade de Washington [Schmidt 97]. Em ambientes deterministas e com carga delimitada, esse ORB pode oferecer garantia de natureza determinista permitindo a execução de aplicações de tempo real *hard*.

A linguagem de descrição de interfaces (IDL) é estendida para permitir a descrição de restrições temporais associadas com cada método nas declarações de interfaces, tal como o tempo máximo de computação. Um *serviço de escalonamento de tempo real* é implementado e funciona em parte de modo *off-line* e em parte *on-line*. Nas versões atuais do TAO, a política de escalonamento usada é o *rate monotonic* que considera as *threads* (métodos) como tarefas independentes. As informações de tempo real especificadas na IDL são utilizadas para um teste de escalonabilidade executado em tempo de projeto, resultando em prioridades armazenadas numa estrutura que é consultada pelo serviço de escalonamento. Em tempo de execução, uma *thread*, criada para tratar uma invocação, tem uma prioridade atribuída pelo adaptador de objetos CORBA a partir de interações com o serviço de escalonamento.

O TAO também apresenta um *serviço de eventos de tempo real* conforme os requisitos descritos na seção 2.2. Além disso, um novo protocolo de interoperabilidade, o RIOP, é definido para suportar aplicações com características *hard real-time*.

4. Propostas de padronização

Em setembro de 1997 a OMG publicou o "edital" RFP (*Request for Proposal*) [OMG 97] solicitando propostas de padronização. Em janeiro de 1998 cinco propostas [Northern 98, Alcatel 98, Objective 98, Lockheed 98, Highlander 98], resultado da materialização de dois anos de discussões, foram apresentadas. Essas propostas deverão compor uma única proposição que terá a sua consolidação como padrão no final de 1998.

Dentre as cinco propostas apresentadas, sem dúvida a [Alcatel 98] é a mais completa, abrangendo interfaces que permitem desde abordagens deterministas até abordagens dinâmicas. A apresentação das propostas de padronização, portanto, destacará inicialmente as principais características dessa primeira. Em seguida, as características mais importantes das outras serão ressaltadas a partir de suas diferenças em relação a essa primeira proposta.

Proposta [Alcatel 98]

Essa proposta enfatiza a definição de mecanismos ao invés de políticas. O motivo é que a especificação de um seleto grupo de políticas dificilmente irá satisfazer todas as aplicações de tempo real. A definição de mecanismos, por outro lado, permite eles sejam utilizados para compor políticas específicas em cada aplicação.

O principal mecanismo fornecido pelo ORB é o *interceptador*[2] (*interceptor*) que permite se construir ambientes de execução adaptáveis às necessidades particulares de uma aplicação. Interceptadores são objetos de aplicação que têm várias de suas operações chamadas pelo ORB em diferentes instantes de uma invocação. Um interceptador no cliente é ativado quando uma invocação é feita e quando a resposta é recebida do servidor. O interceptador no cliente pode ser usado para monitorar as invocações de métodos, ou para passar informações de QoS de forma transparente à aplicação. Um interceptador no servidor é ativado quando uma invocação é recebida e quando a resposta é enviada. A interceptação no servidor pode ser usada para extrair as informações de QoS passadas junto com a invocação, e no controle das semânticas da invocação que decorre dessas informações. Implementações de protocolos de meta-objetos [Maes 87] que separam os aspectos funcionais da aplicação do controle de seu comportamento, podem ser realizadas com o uso desses interceptadores. É possível criar outras categorias de interceptadores que atuariam, por exemplo, junto ao adaptador de objeto, à camada de transporte ou na gestão de *threads*. Interceptadores também podem ser usados num serviço de prioridade global ou para implementar polimorfismo de desempenho.

Threads, grupo (*pool*) de *threads* e filas de requisições são alguns recursos que aplicações podem reservar e manipular diretamente. Por exemplo, um grupo de *threads* pode ser criado para tratar as requisições (invocações de métodos) recebidas por um determinado servidor. Uma fila de requisições pode ser criada e associada com esse grupo de *threads*. A ordenação nessas filas pode ser especificada como FIFO, baseada nas prioridades globais, ou ainda qualquer outra política especificada na aplicação e implementada através do mecanismo de interceptadores.

A noção de *binding* diz respeito ao estabelecimento de uma associação (um caminho de comunicação) entre objetos. O estabelecimento de *binding* explícito entre objetos clientes e servidores, é também coberto pela proposta [Alcatel 98]. Do lado do cliente o *binding* pode ser usado para garantir que seus parâmetros de QoS especificados sejam respeitados, através, por exemplo, da seleção de protocolos de comunicação apropriados. Do lado do servidor, o *binding* permite que se estabeleça os recursos necessários, tais como *threads* e *filas* que serão usados nas execuções das requisições.

Um adaptador de objetos para tempo real (RT POA) é proposto, estendendo o adaptador de

[2] Interceptadores, inicialmente especificados no serviço de segurança do CORBA, são implementados em alguns ORBs comerciais como o ORBIX™ e Visibroker™, e recentemente (fevereiro de 1998) foram padronizados pela OMG no CORBA v2.2.

objetos portável (POA) especificado pela OMG. Essa extensão é necessária para prevenir que o adaptador de objetos execute ações que não permitam previsibilidade, tal como, fazer chamadas externas a outros componentes CORBA, ou procurar implementações de objeto dinamicamente durante uma invocação.

Na proposta [Alcatel 98] é definido um serviço de escalonamento baseado em prioridades fixas. Através das interfaces desse serviço essas prioridades são atribuídas às entidades escalonáveis (usualmente *threads*). Essas prioridades atribuídas podem também ter seus valores lidos posteriormente. O serviço de escalonamento implementa o protocolo de *ceiling* [Sha 90] para delimitar as inversões de prioridades. Nessa primeira submissão não foram propostas extensões aos serviços CORBA de controle de concorrência e de eventos, apesar de ressaltada no texto a importância de que esses serviços sejam devidamente estendidos.

Proposta [Northern 98]

Essa proposta, diferentemente da proposta anterior, visa apresentar um conjunto mínimo de serviços. Algumas das características importantes, necessárias às aplicações de tempo real (manipulação de prioridades em *threads*, por exemplo), não são cobertas pois supõe-se a utilização do suporte oferecido pelo sistema operacional de tempo real subjacente. Para implementar políticas dinâmicas ou políticas específicas a algumas aplicações, essa proposta também sugere a utilização de interceptadores. Entretanto, diferente da proposta anterior, nenhuma interface específica ou mecanismo é apresentado para implementar esse conceito.

Da mesma forma que na proposta anterior, é apresentada uma interface para *binding* explícito. Entretanto, a interface apresentada é bem mais simples e preocupa-se exclusivamente com o lado do cliente. Dentre as características que podem ser configuradas nesse *binding* ressalta-se o valor de um *timeout* para detecção de falhas em invocações, e o tamanho da fila de requisições no servidor. Um serviço de notificação assíncrona de eventos é introduzido, porém sem especificar como indicar restrições temporais nos tratamentos associados às ocorrências dos eventos.

Proposta [Lockheed 98]

Apesar de apresentar em quase todo texto discussões sobre abordagens de escalonamento, essa proposta só apresenta interfaces que atendem a abordagens de escalonamento estático. Interfaces para políticas dinâmicas não são apresentadas sob a argumentação que essa questão será tratada mais tarde pela OMG. Essa proposta apresenta um conjunto de serviços envolvendo a gestão de recursos e o escalonamento de tempo real. A configuração do suporte de comunicação é feita de forma flexível com o uso de *binding* explícito.

O serviço de escalonamento define diversos métodos para atribuir e obter prioridades de objetos (*threads*). Um *serviço de eventos de tempo real* é proposto através de uma interface que permite atribuição de prioridades a objetos associados a eventos.

Proposta [Highlander 98]

Essa proposta, assim como a apresentada a seguir, tem uma preocupação básica em definir interfaces que permitam um escalonamento baseado em prioridade fixas e um protocolo de *ceilling*. O adaptador de objetos POA é estendido, no sentido de se obter um RT POA, onde é possível se manipular as prioridades de *threads* para a implementação da política de escalonamento. Uma interface para *binding* explícito entre objetos também é proposta, entretanto nessa primeira submissão, esta não é apresentada de forma completa.

Proposta [Objective 98]

Essa proposta é a mais simples de todas, apresentando apenas um pequeno número de interfaces basicamente responsáveis por lidar com prioridades de *threads* e mecanismos de sincronização tais como barreiras, *mutexes* e semáforos. Em todos os mecanismos de sincronização é possível atribuir um valor de prioridade para ser usado por um protocolo de *ceiling*. Com relação às *threads*, uma interface especificada permite estabelecer se a política a ser usada será baseada em prioridades ou em prioridades com tempo delimitado (*time-sliced*).

5. Conclusões

As diversas propostas apresentadas refletem diferentes grupos com diferentes perspectivas em relação à padronização do CORBA RT. As abordagens em [Objective 98] e [Highlander 98] se preocupam apenas em estabelecer um CORBA RT "mínimo", com mecanismos simples para "controlar a alocação de recursos" e para "ordenar a execução de tarefas por prioridades". Essas propostas estão mais preocupadas com as questões indicadas nos requisitos para o ambiente operacional descritos na seção 2.2. Já as abordagens mais completas como [Alcatel 98] e [Northern 98] objetivam uma "previsibilidade fim a fim", preenchendo a maior parte dos requisitos propostos pelo documento RFI da OMG (seção 2.2). A proposta em [Alcatel 98], engloba pesquisas complementares envolvendo previsibilidade em tempo de projeto [Schmidt 97] e abordagens dinâmicas [Wolfe 97], dessa forma apresentando um conjunto de interfaces bem mais abrangente e flexível.

Uma outra questão que deve se levar em consideração, é o surgimento recente de diversas pesquisas envolvendo CORBA com aplicações de tempo real, em abordagens dinâmicas e flexíveis. Em [Kalogeraki 97] é apresentado o sistema Realize que se utiliza do CORBA para implantar a noção de objetos distribuídos tolerantes a faltas. Esse sistema possui um gerente de recursos, implementado com o uso de interceptadores, que monitora e controla a utilização dos recursos. Em [Gergeleit 97] a noção de interceptadores CORBA é usada novamente, através do mecanismo de filtros definidos no ORBIX™, para monitorar invocações a objetos CORBA, e o resultado desse monitoramento é usado para realimentar uma abordagem de escalonamento adaptativo. Em [Furtado 96] e [Cooper 97] é descrito o uso de protocolos meta-objetos sobre plataformas CORBA, que são configurados para permitir que aplicações controlem suas funcionalidades de acordo com seus requisitos temporais. Mecanismos de monitoramento e esquemas adaptativos semelhantes ao polimorfismo de desempenho, também são usados em [Cooper 97]. Em [Feng 97] é apresentado uma proposta para permitir que objetos servidores CORBA tratem requisições segundo políticas dinâmicas, e que os clientes possam receber garantias efetuadas dinamicamente através de um teste de admissão. Todas essas pesquisas preconizam a necessidade de especificação de mecanismos (interceptadores, por exemplo) que permitam às aplicações implementarem suas políticas específicas.

A especificação final do CORBA RT irá apresentar diversas abstrações (interfaces e mecanismos) suficientes para execução de aplicações de tempo real. Essas abstrações permitirão que uma variedade de modelos de programação para tempo real possam ser construídos — desde modelos deterministas, até modelos *best-effort* ou adaptativos. O estudo descrito nesse texto é parte de um trabalho que vem sendo conduzido no LCMI/UFSC com objetivo desenvolver uma abordagem de escalonamento adaptativa para aplicações distribuídas de tempo real [Montez 97]. O ambiente de aplicação considerado tem características dinâmicas, prevendo a existência de métodos (tarefas) *hard* e *soft*.

O modelo proposto possui características adaptativas no sentido que algumas decisões de escalonamento se fundamentam sobre condições observadas do sistema através de monitoramentos. A abordagem de escalonamento vem sendo construída fazendo uso de serviços e conceitos CORBA RT. O conceito de interceptador, por exemplo, deverá ser usado para monitorar invocações e implementar técnicas de adaptação. Como forma de adaptação, o polimorfismo de desempenho, e algumas outras técnicas, como o controle do tempo de execução e da freqüência de ativação das tarefas periódicas, estão sendo consideradas. Inicialmente, o trabalho está tomando como base os requisitos descritos no documento RFI da OMG e a experiência do TAO (seção 3.2). O ambiente de execução considerado é constituído por receptores GPS, redes ATM e padrões POSIX para o sistema operacional.

6. Referências

[Alcatel 98] OMG, *Realtime CORBA - Version 1.0*, Initial RFP Submission, Alcatel, HP, Lucent, OOC, Sun, Tri-Pacific, Document orbos/98-01-08, Jan. 1998.

[Cooper 97] G. Cooper, et. al., *Real-Time CORBA Development at MITRE, NRaD, Tri-Pacific and URI*, Proc. of 1st IEEE WMDRTSS, San Francisco, CA, Dec. 1997.

[Feng 97] W. Feng, U. Syyid, J. W.-S Liu, *Providing for an Open Real-Time CORBA*, Proc. of 1st IEEE WMDRTSS, San Francisco, CA, Dec. 1997.

[Furtado 96] 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, Canela, Brasil, 1996.

[Gergeleit 97] M. Gergeleit, E. Nett, M. Mock, *Supporting Adaptive Real-Time Behavior in CORBA*, Proc. of 1st IEEE WMDRTSS, San Francisco, CA, Dec. 1997.

[Highlander 98] OMG, *Realtime CORBA, Joint Initial Submission*, Highlander Communications, Visigenic Software, Inc., Jan. 1998.

[Kalogeraki 97] V. Kalogeraki, P. M. Melliar-Smith, L. E. Moser, *Soft Real-Time Resource Management in CORBA Distributed Systems*, Proc. of 1st IEEE WMDRTSS, San Francisco, CA, Dec. 1997.

[Lockheed 98] OMG, *Realtime CORBA, Response to OMG RFP for Realtime CORBA Extensions*, Lockheed Martin Federal Systems, Inc., Document orbos/98-01-04, Jan. 1998.

[Maes 87] P. Maes, *Concepts and Experiments in Computational Reflection*, Proc. of OOPSLA'87, 1987.

[Montez 97] C. Montez, *Um Modelo de Programação para Aplicações de Tempo Real em Sistemas Abertos*, Monografia: exame de qualificação de doutorado, LCMI-UFSC, Jul. 1997.

[Northern 98] OMG, *Realtime CORBA Extensions*, Joint Initial Submission, Northern Telecom, Iona Technologies, Document orbos/98-01-09, Jan. 1998.

[Objective 98] OMG, *Realtime CORBA, Initial Submission*, OIS, Inc., Jan. 1998.

[OMG 95] OMG, *The Common Object Request Broker: Architecture and Specification - Revision 2.0*, Object Management Group (OMG), Jul. 1995.

[OMG 96] OMG Realtime Platform SIG, *Realtime Technologies - RFI*, Object Management Group (OMG), Document realtime/96-08-02 96, Aug. 1996.

[OMG 96b] OMG Realtime Platform SIG, *Real-Time CORBA - White Paper - Issue 1.0*, Object Management Group (OMG), Dec. 1996.

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

[OMG 97b] OMG, *CorbaServices: Common Object Service Specification*, Object Management Group (OMG), Revised Edition, Mar. 1997.

[Schmidt 97] D. C. Schmidt et al., *TAO: A High Performance Endsystem Architecture for Real-Time CORBA*, IEEE Communications Magazine, 14(2), Feb. 1997.

[Sha 90] L. Sha, R. Rajkumar, J. Lehoczky, *Priority Inheritance Protocols: An Approach to Real-Time Synchronization*, IEEE Trans. on Computers, Vol. 39, No. 9, Sept. 1990.

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

[Wolfe 97] 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.

[Yang 98] Z. Yang, C. Sun, *CORBA for Hard Real Time Applications: Some Critical Issues*, 1st ISORC, Kyoto, Japan, Apr. 1998.