Um Modelo de Programação para Aplicações de Tempo Real em Ambientes ODP

Adilson Barboza Lopes[1]
Alberto Barbosa[2]
Rosa Cristina Martins de Medeiros[3]
Maurício Ferreira Magalhães[3]
{adilson,abarbosa,rosa,mauricio}@dca.fee.unicamp.br
Dep. Eng. Computação e Automação Industrial - FEE
Universidade Estadual de Campinas/UNICAMP
C.P. 6101 CEP 13081-970
Fax (0192) 391395
Campinas/SP

Resumo

Este trabalho apresenta a proposta de um modelo de programação distribuída orientado a objetos para aplicações de tempo real em ambientes ODP. Na concepção deste modelo buscou-se contemplar uma classe de aplicações cujos requisitos temporais admitem certos níveis de tolerância e flexibilidade em relação a eventuais atrasos, como por exemplo, aplicações CSCW(**Computer-Supported Cooperative Work**) e multimídia. Nossa proposta incorpora os benefícios do uso do paradigma de programação orientada a objetos e as regras de estruturação recomendadas pelo modelo básico de referência RM-ODP no que se refere às linguagens de engenharia e computacional. Uma aplicação no modelo consiste de objetos básicos de engenharia configurados segundo os conceitos de cluster, cápsula e nó. Os objetos são componentes passivos enquanto que as atividades são modeladas segundo os conceitos tradicionais de **threads**. O escopo de gerenciamento de uma atividade está limitado ao seu cluster.

[1]UFRN-CCE-DIMAP. Natal.
[2]PUCAMP - Instituto de Informática. Campinas.
[3]UNICAMP - FEE-DCA. Campinas

Chamadas de operações externas ao cluster são realizadas através de canais de comunicação; estas requisições são processadas com semântica tipo **cliente/servidor**. No modelo os requisitos de tempo e qualidade são atribuídos às atividades, clusters e cápsulas. Para suportar tolerância a falhas e prover características de adaptação dinâmica mediante a ocorrência de situações imprevistas, o modelo incorpora o conceito de Qualidade de Serviço(**QoS**). A contribuição deste trabalho consiste em introduzir o conceito de função benefício em conjunção com múltiplas versões de implementação de objetos como suporte ao tratamento de Qualidade de Serviços(**QoS**).

Abstract

This work proposes a distributed real time object-oriented programming model in ODP context. In this model we are interested in a class of application whose time requirements tolerate shorts levels of delay as multimedia and **CSCW**(Computer Supported Cooperative Work) applications. In order to support the requirements of this kind of application, our proposal have been considered the Object Oriented Program Paradigm and the recommendations of the Basic Reference Model RM-ODP, specially those which are concerned with the engineering and computation viewpoints. An application in the model consists of objects whose definition structures and instances are realized in according with the RM-ODP concepts of Basic Engineering Object, Clusters, Capsule and Node. The objects are passive entities in which system activities flow and activities are abstracted from traditional concept of threads. Activities scope is limited to its clusters frontiers. Invocation of remote operations are realized transparently by communication channels. The semantics of remote interactions is based on client/server model paradigm. Information about predicted execution time, deadline, requirements of quality, etc. are bound with activities. In order to support fault tolerance and to improve flexibility features in the sense that dynamic system adaptations are supported without affecting temporal requirements, the model introduce the Quality of Service(QoS) concept. The contribution of this work is to integrate benefit function and multiversion concepts as a mechanism to handle Quality of Services.

1 INTRODUÇÃO

Os trabalhos desenvolvidos nos últimos anos na área de sistemas de tempo-real foram caracterizados pela implementação de núcleos de tempo-real e modelos/linguagens de programação/especificação. Entretanto estes trabalhos não deram a ênfase necessária ao tratamento da variável tempo no sentido de permitir a estes sistemas

um comportamento determinístico quando da sua execução. A questão atualmente colocada nas pesquisas voltadas para a área de tempo-real é a de que os requisitos desses sistemas não se restringem apenas em garantir a correção funcional da aplicação, mas também, que ela seja executada no tempo apropriado. Exemplos típicos dessas situações ocorrem no controle do núcleo de um reator, controle de tráfego aéreo, controle de aeronaves, controle de um conjunto de robôs trabalhando num chão de fábrica, aplicações multimídia,etc.

A evolução tecnológica que vem ocorrendo nos últimos anos está popularizando cada vez mais o uso de sistemas distribuídos de computação; atualmente, com a instalação de redes de fibra ótica e de novas técnicas de comunicação de alto desempenho, tem-se criado uma base tecnológica para suportar a existência de aplicações cooperativas em sistemas distribuídos de grande porte. Estas aplicações envolvem uma diversidade de requisitos, equipamentos, linguagens de programação e fornecedores.

Com o objetivo de estabelecer uma arquitetura que permita o desenvolvimento e a programação de aplicações distribuídas sem considerar a potencial diversificação de **hardware**, sistemas operacionais e mecanismos de comunicação, a ISO e a ITU-T estão desenvolvendo um Modelo de Referência Básico para Sistemas Abertos de Processamento Distribuído(RM-ODP[2,3,4,5]).

Diante desse cenário o nosso trabalho apresenta uma proposta de um modelo de programação orientado a objetos para desenvolvimento de sistemas distribuídos de tempo-real. O modelo incorpora os benefícios do uso do paradigma de Programação Orientada a Objetos(**POO**) e as regras de estruturação recomendadas pelo Modelo RM-ODP no que se refere às linguagens de engenharia e computacional.

Com o objetivo de melhor caracterizar as atividades componentes de uma aplicação de tempo real a literatura tem consolidado uma abordagem em que as aplicações são formadas por tarefas que possuem restrições críticas de tempo(**hard**) e que devem executar o mais rapidamente possível, mas que não necessitam cumprir prazos(**soft**). Porém esta taxonomia tem se mostrada inadequada para a maior parte das aplicações de tempo real uma vez que a grande complexidade destes sistemas pode tornar os requisitos de garantia de correção temporal um problema computacionalmente intratável.

A complexidade dos sistemas de tempo real é aumentada quando a aplicação envolve um ambiente de processamento distribuído aberto(ODP) onde alguns componentes podem estar interconectados por meio de diferentes subredes e pertencer a vários domínios administrativos. Nestes sistemas a maior parte dos componentes é instalada, operada e usada de acordo com os requisitos locais; dessa forma os componentes são caracterizados por diferentes políticas de gerenciamento, condições de operação, disponibilidade, carga, etc. Assim cada componente tem autonomia restrita ao seu ambiente local, com comportamento determinado pelas atividades e políticas de gerenciamento locais. Esta visão de sistemas distribuídos é conflitante

com a abordagem clássica de sistemas de tempo real,no que se relaciona, por exemplo, com a garantia de prazos no processamento de atividades **hard** que dependam da execução de serviços remotos, cujas características e domínio de gerenciamento são totalmente independentes.

Para contornar estes conflitos e objetivando prover características de tolerância a falhas e adaptação a mudanças dinâmicas do sistema, o nosso trabalho propõe a integração dos mecanismos de função benefício[6,7,8] e múltiplas versões[1,9]. Através dos mecanismos de múltiplas versões o modelo suporta os conceitos de Qualidade de Serviços(**QoS**)[8,19,21], onde múltiplas versões de implementação representam diferentes níveis de qualidade de serviços a serem negociados em tempo de execução, levando-se em consideração o comportamento dinâmico das funções benefício dos componentes do sistema. Esta solução envolve um compromisso a ser estabelecido entre os aspectos de autonomia local e requisitos de tempo real num ambiente ODP.

Este artigo está estruturado da seguinte forma: a seção 2 faz uma abordagem sobre as características das aplicações de tempo real atuais; a seguir, a seção 3 apresenta o modelo proposto neste trabalho, enquanto que, a seção 4 faz uma rápida discussão sobre o suporte de programação para o modelo. Finalizando, a seção 5 apresenta as conclusões do trabalho e as perspectivas futuras.

2 Uma Abordagem sobre as Aplicações de Tempo Real Atuais

Nos últimos anos os sistemas de tempo real têm se expandido rapidamente. Em particular, a classe de aplicações CSCW envolvendo sistemas de computação multimídia tem se tornado uma realidade presente em diversas atividades, como por exemplo, teleconferências, elaboração cooperativa de documentos, sistemas de suporte multimídia para desenvolvimento de programas educacionais, etc. Entretanto estes sistemas constituem uma classe de aplicações que apresentam algumas diferenças em relação aos conceitos tradicionais de sitemas de tempo real[8]:

- As restrições de tempo dependem da **QoS** dos serviços que estão sendo requisitados, como por exemplo, o número de quadros por segundo na transmissão de uma imagem, os requisitos de cor e atrasos no envio de informações de imagens, etc;
- A carga do sistema é fortemente dependente do tempo e estado do sistema;
- A violação de requisitos de tempo pode ser tolerada por períodos curtos de tempo, implicando, neste caso, num impacto negativo na qualidade do serviço que está sendo oferecido;

- O ambiente deve ser aberto, permitindo desta forma a entrada e saída dinâmica de provedores e clientes de serviços.

O desenvolvimento deste tipo de aplicação se depara com novos problemas, como, por exemplo, o envolvimento simultâneo de diferentes meios de comunicação, os quais impõem restrições diferenciadas para os tipos de informações transmitidas(áudio, vídeo, texto, gráficos, etc.). Estas restrições incluem limitações de atrasos, garantia de eficiência nos canais de comunicação, etc.

Outro fato importante a ser ressaltado é que a nova geração de aplicações distribuídas deve ser orientada por um modelo de referência padrão; entretanto o estado atual do Modelo Básico de Referência RM-ODP não tem contemplado todos os requisitos deste tipo de aplicação. Nesse sentido alguns trabalhos recentes têm se preocupado em incrementar o modelo ODP com o intuito de prover suporte às aplicações multimídia. Nestes trabalhos o conceito de **QoS** tem se apresentado como a melhor forma para expressar os requisitos de multimídia[21]. O desafio consiste em incorporar estes conceitos aos vários níveis de abstração do modelo de referência RM-ODP, bem como analisar as suas implicações para os requisitos de comunicação.

A seguir são apresentadas algumas propostas discutidas na literatura que incorporam os conceitos de **QoS**:

Em [21] é apresentada uma proposta, com ênfase para a visão de engenharia, que consiste em introduzir objetos para gerenciamento de qualidade de serviços nos nós do sistema; estes objetos permitem o gerenciamento de **QoS** em ambientes ODP a partir das informações fornecidas através de declarações de atributos e qualidade de serviços correspondentes aos meios de comunicação e interfaces de serviços. A idéia consiste em manter canais de informações de gerenciamento ligando os diferentes objetos de gerenciamento de **QoS**; estes canais permitem a comunicação de informações de gerenciamento, as quais são necessárias para informar aos diversos gerentes sobre os requisitos de **QoS** requeridos pelo sistema e as mudanças dinâmicas ocorridas que venham a implicar em degradações toleráveis nos resultados do sistema. As funções básicas de um gerente de **QoS** são definir os níveis de **QoS** para a execução de serviços e selecionar um protocolo de comunicação adequado.

A proposta apresentada em [19] consiste em estender o modelo ODP através da introdução de objetos de tempo real reativos e da declaração de comandos qualificadores de serviços. Desta forma as idéias apresentadas em [19] preservam o modelo computacional da proposta ODP. Nessa proposta uma atividade consiste de um conjunto de objetos que interagem entre si, alguns dos quais caracterizados como objetos reativos; neste caso, os requisitos de tempo real devem ser assegurados. Objetos não reativos não se preocupam com aspectos de garantia de requisitos de tempo real. Neste modelo as restrições temporais são isoladas a nível de declarações de **QoS**; usando estas informações, o gerenciador de **QoS** se encarrega de controlar os recursos requisitados pelo sistema de comunicação e pelos objetos da aplicação. As

operações, a nível de negociação e renegociação de **QoS**s, são tratadas segundo requisitos de garantia, ou, na impossibilidade de assegurar a garantia, usando técnicas do tipo melhor esforço; o gerenciador é também responsável por preservar níveis de **QoS** contratados anteriormente.

Em [20] é apresentada uma nova classe de serviços de comunicação de tempo real que consiste em interagir com a rede de comunicação no sentido de especificar e negociar **QoS**s de conexões. Este modelo busca prover serviços de **QoS**s adaptáveis às mudanças de estado da rede de comunicação em face a situações de sobrecarga e congestionamentos temporários. Assim, no estabelecimento de uma requisição, o modelo, ao invés de impor parâmetros rígidos de atraso, deve expressar os seus requisitos em termos de variações de qualidade e eficiência. A idéia consiste em tolerar níveis aceitáveis de degradação no processamento e no suporte de comunicação. Esta solução é baseada em três princípios: melhorar a qualidade das informações a serem fornecidas para o cliente quando a comunicação for rejeitada; melhorar a eficiência para estabelecimento do canal; e por fim, introduzir uma classe de serviços de comunicação mais flexível.

O ítem a seguir apresenta a proposta de um modelo para programação de aplicações de tempo real; esta proposta procura contemplar algumas das idéias discutidas acima segundo uma visão ODP. A contribuição que este trabalho se propõe realizar consiste em introduzir os conceitos de função benefício[6,7,18] e múltiplas versões[1,9] como mecanismos de suporte ao gerenciamento dinâmico de **QoS** em sistemas de tempo real. O conceito de função benefício se apresenta como um mecanismo bastante poderoso para expressar níveis e limites de degradação de qualidade de processamento, que, juntamente com os conceitos de múltiplas versões, se adequam a esta nova classe de sistemas de tempo real.

3 O Modelo de Programação Proposto

O Modelo de programação proposto neste trabalho é baseado nos conceitos de programação orientada a objetos e tem como objetivo principal prover características mais flexíveis ao processo de desenvolvimento, programação e execução de sistemas distribuídos de tempo real. A decisão de adotar o paradigma de POO reflete uma tendência constatada na literatura onde diversos trabalhos envolvendo ambientes de programação para sistemas de tempo real, desenvolvidos recentemente, têm adotado este paradigma. Na realidade POO tem se consolidado como um modelo adequado para diferentes áreas; isto se deve ao fato de que a maior parte dos objetos do mundo real podem ser virtualmente modelados num sistema POO.

Para suportar portabilidade nossa proposta leva em consideração as recomendações do modelo RM-ODP. O Modelo Geral de Referência RM-ODP define uma estrutura de padronização para sistemas distribuídos abertos. Neste modelo uma aplicação consiste de grupos fracamente acoplados de atividades cooperativas que

trabalham juntas de maneira a obterem um resultado desejado. Visando lidar com a complexidade destes sistemas, o Modelo RM-ODP define cinco pontos de vista que compõem a especificação de aplicações ODP e prescrevem um conjunto de funções de suporte. A especificação de vários pontos de vista de um sistema ODP, onde cada um representa uma abstração diferente do sistema, permite a verificação de completeza e a realização de consistências entre as diferentes visões do sistema. O modelo RM-ODP prevê os seguintes pontos de vista:

- **empresarial** - fornece um modelo para a definição dos papéis e políticas de gerenciamento da empresa;
- **de informação** - define um modelo para especificação e análise de estruturas e fluxos de informações;
- **computacional** - define um modelo de linguagens de programação distribuída com requisitos de transparência em relação a arquitetura. Sob o ponto de vista computacional uma aplicação é caracterizada por uma especificação funcional que define os objetos do sistema, as atividades relacionadas, os tipos de interações que ocorrem entre os objetos, as restri,ões relativas ao ambiente e semântica para tratamento de falhas. Estes conceitos são suportados por regras de estruturação que asseguram os requisitos de transparência de distribuição.
- **de engenharia** - descreve a organização de uma infra-estrutura abstrata de suporte **run-time** para execução de sistemas distribuídos. Sob o ponto de vista de engenharia as entidades básicas que constituem um sistema são objetos e canais. Os objetos podem ser caracterizados como objetos básicos de engenharia(correspondentes aos objetos da especificação computacional) e objetos de infra-estrutura(protocolos, **stubs**, etc.);
- **de tecnologia** - descreve a implementação(configuração, instalação e manutenção de **hardware** e **software**).

O modelo apresentado neste artigo é baseado nas recomendações relativas às visões de engenharia e computacional.

3.1 O Modelo de Objetos

Em nosso modelo uma aplicação consiste de um conjunto de atividades e objetos definidos, instanciados e estruturados de acordo com os conceitos de objeto básico de engenharia, cluster, cápsula e nó[3,4,5]. Em nossa proposta os objetos são entidades passivas por onde fluem as atividades do sistema; já uma atividade consiste de um um fluxo de controle(**thread**) que percorre vários objetos. Diferentes atividades de

um cluster podem ter acesso a um mesmo objeto, ficando o controle de concorrência sob responsabilidade do usuário. As restrições relacionadas com tempo de execução, prazos e qualidade de resultados são atributos das atividades. Uma atividade se restringe ao escopo do seu cluster; assim, a chamada de um método de um objeto externo ao cluster é realizada via canal de comunicação. Esta requisição é transportada para o cluster destino onde será tratada de acordo com o estado e políticas de gerenciamento deste cluster e dos atributos da atividade fonte. Desta forma a semântica de interações remotas é baseada no paradigma **cliente/servidor**.

3.2 Múltiplas versões de atividades

O Modelo proposto provê características de flexibilidade ao suportar o conceito de múltiplas versões de implementação para os objetos e atividades; estas versões podem ser caracterizadas, por exemplo, por possuírem diferentes níveis de performance, qualidade e precisão. Neste trabalho o conceito de múltiplas versões é implementado a nível de objeto. Nossa intenção é manter uma estrutura semântica similar aos mecanismos de polimorfismo suportados pelas linguagens C++[11] e Smaltalk[10].

O Modelo assume que os tempos máximos de execução dos métodos são conhecidos/previsíveis; desta forma é possível prever tempos de execução máximos para cada atividade através das somas dos tempos de execução previstos para cada método componente da atividade. Esta suposição implica em restringir o uso de iterações ilimitadas e chamadas recursivas de métodos na linguagem de suporte ao modelo de programação. Em tempo de execução o suporte do modelo pode atuar na execução das atividades através da escolha de versões em função do estado do sistema. Assim, se a dinâmica da aplicação leva o sistema a um estado de sobrecarga temporária, o suporte deve selecionar versões de atividades mais simples na tentativa de satisfazer os requisitos temporais da aplicação. Estas idéias envolvendo requisitos de qualidade e precisão são derivadas dos conceitos de computações imprecisas[17].

3.3 Função Benefício

Função benefício consiste numa generalização do conceito de **Time Value Function** definido em [18] onde são agrupados os atributos relativos a urgência e importância: a componente urgência considera os requisitos temporais da atividade; já a componente importância considera outros parâmetros que refletem o estado da aplicação e do sistema envolvidos na disputa de recursos.

Uma função benefício estabelece alguns pontos(instantes) especiais que caracterizam marcos cujas informações podem ser usadas para fins de escalonamento(Figura 1):

- t_R - instante a partir do qual a atividade pode ser escalonada(**release time**);

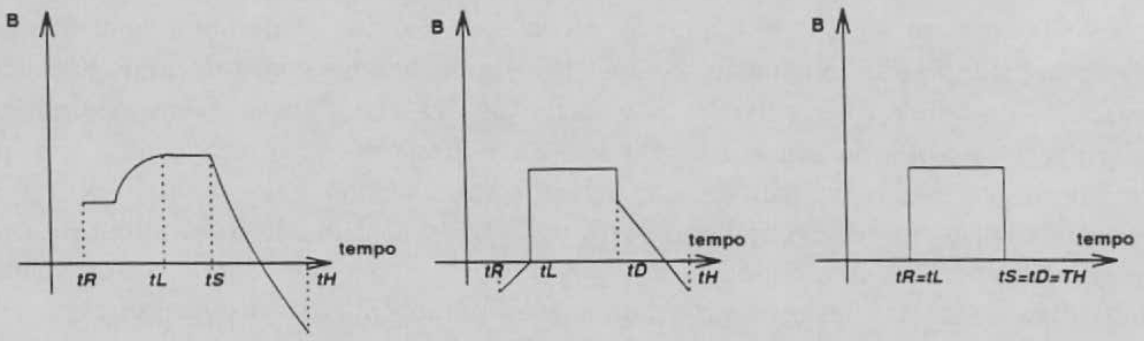

Figura 1: Exemplos de funções benefício

- t_H - instante a partir do qual a atividade não faz mais sentido para o sistema;
- t_L - instante a partir do qual a contribuição da atividade para o sistema deixa de ser crescente;
- t_S - instante a partir do qual a contribuição da atividade para o sistema passa a ser decrescente;
- t_D - instante a partir do qual a execução da atividade provoca uma falha no sistema(**deadline**).

No modelo proposto os requisitos de tempo e qualidade,bem como as contribuições de uma atividade relativas ao sistema são especificadas através de definições de funções benefício: o objetivo é propiciar o aumento do benefício global do sistema. Por exemplo, muitas vezes é melhor para o sistema que uma atividade seja executada, mesmo com um certo atraso, do que arbitrariamente ela seja interrompida no meio do seu processamento. Assim, em nosso modelo as tarefas não são rigidamente classificadas como **hard** ou **soft**; neste caso a função benefício permite expressar vários níveis de caracterização entre **hard** e **soft**.

Considerando que a nossa proposta incorpora os conceitos de cluster e cápsula da linguagem de engenharia do modelo RM-ODP e que o objetivo a ser perseguido consiste em tentar aumentar o benefício global do sistema, o modelo estende o conceito de função benefício para os níveis de cluster e cápsula: a engenharia envolvida na definição dessas funções deve levar em consideração a especificação das várias visões que compõem o sistema ODP e deve ser resultante de uma solução de compromisso envolvendo diferentes políticas de gerenciamento e resultados de análises realizadas na especificação.

A combinação dos conceitos de múltiplas versões e função benefício propiciam características de flexibilidade ao viabilizar a execução de atividades com restrições de tempo rígidas diante de situações de sobrecarga. Considerando que cada versão

de uma atividade possui uma função benefício, o modelo provê, a nível de programação, os requisitos necessários para que o suporte de execução reaja a situações imprevistas, de maneira a afetar o mínimo possível a operação do sistema.

A figura 2 a seguir mostra como o modelo representa os conceitos de atividade **hard** e **soft** usando os mecanismos de múltiplas versões; na figura 2a são representadas 3 versões de uma atividade definida como **hard**: a versão 1 corresponde àquela de melhor qualidade, enquanto que as versões 2 e 3 requerem um tempo de execução menor, mas, em contrapartida, contribuem com um menor benefício para o sistema. No caso de uma atividade **soft**, figura 2b, o modelo considera uma única implementação para os componentes da atividade; assim o que caracteriza cada versão é o instante de término de execução da atividade. Por exemplo, a versão 1 da atividade2 é caracterizada por encerrar seu processamento no instante t_1, provendo, desta forma, o melhor benefício para o sistema. Neste caso, somente ocorrerá falha se o instante t_D for alcançado sem que a atividade tenha sido concluída.

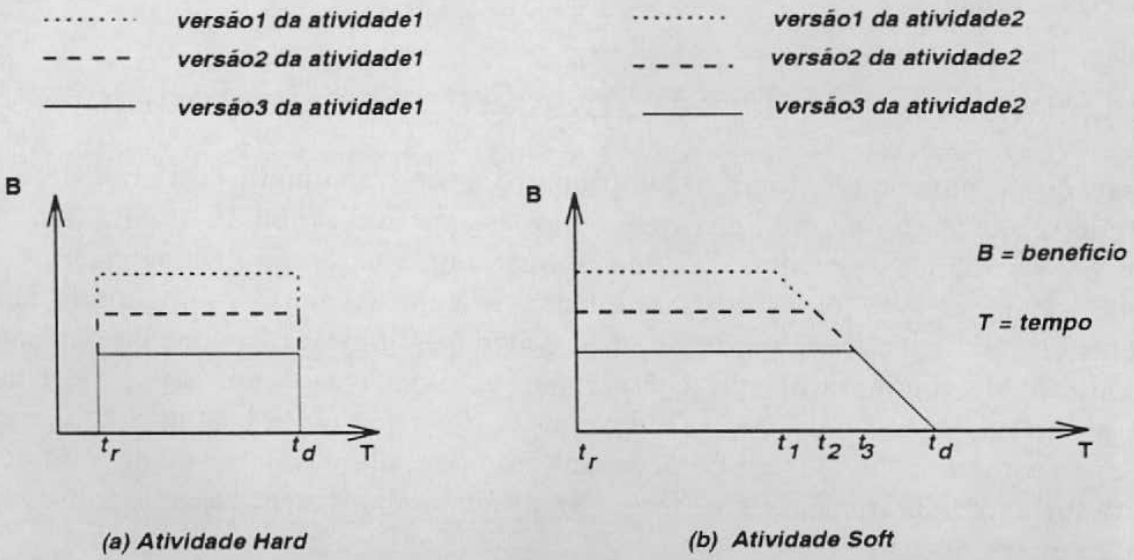

Figura 2: Exemplos de funções benefício

3.4 O Modelo de interação

Todos os objetos no modelo interagem da mesma maneira, ou seja, objetos oferecem serviços através de interfaces cujas regras de definição devem obedecer as recomendações do modelo RM-ODP. Assim a interação ocorre de forma transparente e independente da operação chamada pertencer ao domínio do cluster ou não. Se os objetos cooperantes residirem no mesmo cluster a interação ocorre de forma direta, com comportamento semelhante a uma chamada local de procedimento, respeitadas apenas as restrições de proteção/controle de concorrência especificadas a nível de

objeto. A função benefício associada a cada atividade é monitorada permanentemente pelo gerenciador do cluster a quem cabe implementar a política adequada de escalonamento tempo real.

Uma interação entre objetos remotos é suportada pelo modelo **cliente/servidor**: um objeto cliente requisita uma operação externa ao cluster através de um canal cuja interface é similar a de uma operação local. Este procedimento envolve a preparação da requisição a nível de definição(mapeamento) da função benefício a ser transportada, de maneira a caracterizar os requisitos de tempo e qualidade a serem perseguidos pelo gerenciador do cluster remoto. Esta preparação de requisitos pode envolver desde um processamento simples, tipo uma chamada RPC, até um procedimento bem complexo onde seja necessário uma sequência de adaptações, negociações ou migração de clusters. Caso o gerenciador do cluster servidor não consiga atender a requisição no prazo, o gerenciador do cluster cliente, através de cláusulas de **time-out**, consegue detectar a falha e executar ações de tratamento de exceções adequadas.

4 Suporte de Programação para o Modelo

Baseado no modelo de programação proposto neste trabalho nós estamos desenvolvendo um projeto de uma linguagem de programação distribuída orientada a objetos para sistemas de tempo real que consiste numa extensão da linguagem C++. Esta extensão provê características adicionais à linguagem C++ incluindo função benefício, múltiplas versões de objetos, atividades, objetos básicos de engenharia e objetos de configuração que contemplam os conceitos de nó, cápsula e cluster. O porquê da escolha de C++ se deve ao fato de que ela é bastante conhecida e tem se tornado a linguagem de programação orientada a objetos mais popular nos últimos anos. Os ítens a seguir fazem uma rápida abordagem sobre as construções da linguagem:

- Objeto básico de engenharia - os objetos básicos de engenharia são definidos de maneira idêntica aos objetos C++; as interfaces dos objetos especificam o conjunto de operações e seus atributos. Somente objetos que suportam múltiplas versões devem especificar uma palavra-chave especial na classe base, ao invés da palavra-chave **class** do C++ (figura 3);

- Configuração de Clusters - a configuração de um cluster consiste de uma especificação de uma classe especial(**clusterclass**) que instancia um conjunto de objetos básicos de engenharia e suas atividades(figura 4);

- Configuração de nó e cápsula - o processo de configuração de nós e cápsulas é similar à configuração de clusters: os gerentes de clusters e seus objetos de

```
RTPCLASS exemplo-basico1 {
PUBLIC:
        VIRTUAL VOID operacao1(VOID)
                . . .

PRIVATE :
        INT operacao2(VOID);
                . . .

}

CLASS versao1: PUBLIC exemplo-basico1 {
PUBLIC:
        /* primeira versão de implementação da operação1 */
        VOID operacao1(VOID) {
                . . .
        }
                . . .
PRIVATE:
                . . .
}

CLASS versao2: PUBLIC exemplo-basico1 {
PUBLIC:
        /* segunda versão de implementação da operação1 */
        VOID operacao1(VOID) {
                . . .
         }
                . . .
PRIVATE:
                . . .
}
```

Figura 3: Exemplo de definição de objeto

```
CLUSTERCLASS definicao1 {
PUBLIC:
        /* Instâncias  dos objetos que compoem um cluster */
        exemplo-basico1 objeto1(VOID);
                . . .

PRIVATE:
        /* operaões - objetos internos do cluster */
                . . .

ACTIVITIES:
        /* declaração das atividades de um cluster e especificação */
        /* das funções benefício e atributos/requisitos de qualidade */
        operacao1(VOID) /BENEFIT = 1.0  /RELEASE = 100  /DELAY-MAX-TRANSMISSAO = 100
                        /DEADLINE = 500 /PERIOD = 500 /TAXA-ERRO-TRANSMISSAO < 0.000001
              SUCCESS(resultado) {
                                        . . .
                                    ação sucesso
                                        . . .
              }
              FAIL(resultado-falha) {
                                        . . .
                                    ação falha
                                        . . .
              }
         operação2(VOID)  /beneficio e atributos da operacao
                        . . .
}
```

Figura 4: Exemplo de configuração de cluster

suporte e infra-estrutura são instanciados a nível de configuração de cápsulas; a configuração de um nó consiste de instâncias de cápsulas e de um objeto núcleo.

5 Conclusão

Neste trabalho nós apresentamos uma proposta de um modelo de programação orientado a objetos para aplicações de tempo real distribuídas. Esta proposta leva em consideração as recomendações do modelo básico de referência RM-ODP e busca contemplar uma nova classe de aplicações de tempo real onde são tolerados eventuais atrasos, tais como, aplicações CSCW e multimídia. Atualmente encontra-se em fase de desenvolvimento uma implementação simplificada de um ambiente de suporte ao modelo de programação e escalonamento. Nesta primeira fase a nossa implementação está se limitando ao escopo de gerenciamento de clusters. Paralelamente estão sendo desenvolvidos estudos no sentido de implementar algumas soluções propostas na literatura para gerenciamento de qualidade de serviços no contexto ODP. Nosso objetivo é avaliar estas idéias e validar os conceitos de função benefício e múltiplas versões como mecanismos de suporte ao gerenciamento de **QoS** em ambientes ODP.

Agradecimentos

Os autores agradecem à FAPESP(Projeto Temático N^o 92/3507), ao CNPq e CAPES(Programas de bolsas de mestrado e doutorado e Programa RHAE), bem como a IBM(cessão de equipamentos e software), pelo apoio financeiro e logístico dado ao trabalho de desenvolvimento deste projeto.

Referências

[1] Kenny,K.K and Lin K. Building Flexible Real-Time Systems Using the Flex Language. Computer IEEE, may 1991.

[2] Raymond K.A. Reference Model of Open Distributed Processing: A Tutorial. 2nd International Conference on Open Distributed Processing - ICODP'93. 1993.

[3] ISO/IECJTC1/SC 21/WG7 N183. "Basic Reference Model of Open Distributed Processing - Part 1: Overview and Guide to Use". 1993.

[4] ISO/IEC CD 10746-2.3. "Basic Reference Model of Open Distributed Processing - Part 2; Descriptive Model". 1993.

[5] ISO/IECJC1/SC21/WG7 N8125. "Basic Reference Model of Open Distributed Processing - Part 3: Prescriptive Model". 1993.

[6] Jensen E.D. A Benefit Accrual Model of Real-Time Computing. 10th IFAC Workshop on Distributed Computer Control Systems. 1991.

[7] Jensen E.D., Locke C.D. and Tokuda H. A Time-Value Driven Scheduling Model for Real-Time Operating Systems. Proceedings of 10th IEEE Real-Time Systems Symposium. 1985.

[8] Cardozo E., Magalhães M. and Adán J.M. Programming Flexible Distributed Real-Time Systems. In Proceedings of the fourth IEEE Workshop On Future Trends on Distributed Computing Systems. 1993.

[9] Yau S.S., Bae D. Oh, Chidambaran. An Object-Oriented Approach to Software Design for Distributed Real-Time Computing Systems. In Proceedings of the fourth IEEE Workshop On Future Trends on Distributed Computing Systems. 1993.

[10] Goldberg A. and Robson D. Smalltalk-80, the Language and its Implementation. Addison-Wesley Inc. 1983.

[11] Stroustrup, B. The C++ Programming Language. Addison-Wesley Inc. 1991.

[12] Wu Z. Making C++ A Distributed Programming Language. In Proceedings of the IEEE fourth Workshop On Future Trends on Distributed Computing Systems. 1993.

[13] Takashio K. and Tokoro M. DROL: An Object-Oriented Programming Language for Distributed Real-Time Systems. OOPSLA'92. 1992.

[14] Wang F., Chen J. and Hu C. A Distributed Object-Oriented System with Multi-threads of Services. In Proceedings of the IEEE fourth Workshop On Future Trends on Distributed Computing Systems. 1993.

[15] Birrel A.D. and Nelson B.J. Implementing Remote Procedure Calls. ACM Transactions on Computer Systems, 2(1) February 1984.

[16] Tanenbaum, A.S. Modern Operating Systems. Prentice-Hall Inc.1992.

[17] Liu J.W.S. et al. Algoritms for Scheduling Imprecise Computation. Computer IEEE. May 1991.

[18] Tokuda H. Wendorf J.W and Wang H.Y. Implementation of Time-Driven Scheduler for Real-Time Operating Systems. Proceedings of 12th IEEE Real-Time Systems Symposium. 1987.

[19] Coulson G. et al. Supporting the Real-Time Requeriments of Continous Media in Open Distributed Programming. University of Lancaster Thechinical Report MPG-92-35. 1992.

[20] Ramaekers J. and Ventre G. Client-Network Interaction in Real-Time Communication Environment. University of California Thechinical Report. 1992.

[21] Fedaoui L., Tawbi W. and Horlait E. Distributed Multimedia Systems Quality of Service in ODP Framework of Abstraction: A First Study. 2nd International Conference on Open Distributed Processing - ICODP'93. 1993.