Modelagem de Requisitos de QoS Utilizando UML-RT:

Estudo de caso da especificação de requisitos de QoS para sistemas interativos baseados em redes com o uso de UML-RT

Edison Pignaton de Freitas, Elias Teodoro da Silva Jr, Fabiano Costa Carvalho

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{epfreitas,etsilvajr,fccarvalho}@inf.ufrgs.br

Resumo: A linguagem UML padrão não apresenta todo o ferramental necessário à modelagem de sistemas de tempo-real, mas existe uma extensão para projetos desta natureza que cobre esta deficiência. Para explicitar os requisitos de QoS presentes em sistemas interativos distribuídos em rede, tais como sistemas de vídeo-conferência, pode-se adotar como alternativa a modelagem UML com enfoque de tempo-real. Este trabalho busca apresentar o uso da linguagem UML-RT para modelar aplicações de tempo-real interativas baseadas em redes. Com a apresentação do estudo de caso, procura-se mostrar a utilização de algumas características da UML-RT de forma a conseguir um modelo que expresse com clareza os requisitos de tempo-real pertinentes.

1 – Introdução

Muito se fala sobre o uso de sistemas de tempo-real (TR) para aplicações de automação industrial e sistemas embarcados [EDWARDS 1997]. De fato, estes são os exemplos mais clássicos do uso de sistemas TR. Porém, um outro domínio de aplicação onde o enfoque de tempo-real pode encontrar solo bastante fértil é o dos sistemas interativos distribuídos em rede. Tais sistemas têm uma forte demanda por qualidade de serviço, e apresentam rígidos prazos a serem cumpridos, tanto quanto qualquer outro sistema TR.

O projeto de sistemas de tempo-real apresenta complicadores inevitáveis quanto à definição de seus requisitos mínimos de funcionamento adequado. Isto pode ser atribuído principalmente aos rígidos prazos de execução de tarefas. Num sistema de vídeo-conferência, por exemplo, é indispensável que suas tarefas cumpram os requisitos temporais, caso contrário a interatividade pode ficar comprometida.

A linguagem UML não apresentava, em suas primeiras versões, nenhum suporte às peculiaridades existentes no projeto de sistemas de tempo real. O tratamento destes requisitos somente foi introduzido à linguagem através do perfil UML-RT [OMG 2004], que consolidou uma série de propostas de extensão à linguagem. Em sua última versão, a 2.0, a UML apresenta bem amadurecidas as ferramentas de suporte ao projeto de sistemas de tempo-real.

Este trabalho propõe a utilização dos recursos presentes na UML-RT para a modelagem de sistemas interativos baseados em rede como os de vídeo-conferência. Apesar do uso de redes de Petri estocásticas temporizadas [TSAI, YANG, CHANG 1995] ser bastante adequado à modelagem de sistemas desta natureza, os diagramas UML-RT possuem uma expressividade que merece ser explorada neste domínio.

Na seqüência, a seção 2 apresenta alguns conceitos sobre sistemas de tempo-real, enquanto a seção 3 trata da extensão RT para a UML. Na seção 4 é apresentado um estudo de caso e finalmente na seção 5, as conclusões.

2 – Sistemas de Tempo Real

Sistemas de tempo-real são aqueles que possuem fortes requisitos no que se refere ao atendimento de compromissos de tempo de execução (*deadlines*), robustez e segurança, principalmente. Estas exigências são conseqüência do risco potencial de catástrofes em caso de falha, ou até mesmo no caso do seu funcionamento não ocorrer exatamente como o esperado [DOUGLASS 1999].

Pode-se classificar os sistemas de tempo-real da seguinte forma:

- *Hard Real-Time*: São aqueles nos quais os requisitos temporais devem ser atendidos de forma incondicional. A perda de um prazo (*deadline*) constitui um erro de computação. Neste tipo de sistema, um resultado atrasado é inaceitável.

- *Soft Real-Time*: Neste tipo de sistema os prazos podem ser eventualmente perdidos, ou pode haver um atraso tolerável, ou mesmo as duas hipóteses. Eles são regidos, em sua maioria, por uma média de exigência temporal.

Normalmente os sistemas podem ser classificados em posições intermediárias entre estes dois extremos [SHAW 2001], levando-se em conta o quão rigorosos ou tolerantes à falhas eles são.

Pode-se dizer que um sistema de vídeo-conferência ocupa uma posição intermediária não muito distante do extremo rigoroso (*hard real-time*). Mesmo essa aplicação não sendo tão crítica quanto um controle de tráfego aéreo, por exemplo, a perda freqüente de limites temporais tornaria o sistema completamente inútil, pois não se alcançaria a interatividade desejada.

3 – Extensões da UML para Tempo-Real

Antes do surgimento das extensões UML para sistemas de tempo-real, a linguagem orientada a objetos mais usada neste domínio era a ROOM (*Real-Time Object Oriented Modeling*) [DOUGLASS 1999]. Estudos levaram à proposição do uso da UML na modelagem de sistemas de tempo-real. Porém, esta linguagem ainda não estava apta a representar, com a riqueza de detalhes necessária, as diferentes características estruturais e comportamentais de tais sistemas. Com o intuito de suprir esta deficiência na linguagem UML, foi proposta uma extensão com base em conceitos advindos da linguagem ROOM. Esta proposta resultou num conjunto de conceitos de propósito geral para a modelagem de sistemas de tempo-real. Mais tarde, apareceram outras propostas de extensões à UML para cobertura das necessidades de modelagem no domínio de tempo-real. Tais propostas foram finalmente padronizadas pela OMG através do perfil UML-RT.

O perfil UML-RT apresenta *frameworks* para construção de modelos: 1) de Recursos; 2) de Tempo; 3) da Concorrência; 4) de Análise de Escalonabilidade; e 5) do Desempenho. O foco deste trabalho é apresentar a utilização deste último *framework*, que se propõe a facilitar as seguintes tarefas do projetista: 1) Captura dos requisitos de desempenho no contexto do projeto; 2) Associação das características de QoS relacionadas ao desempenho com elementos específicos do modelo UML; 3) Especificação de parâmetros de execução, que podem ser usados na modelagem de ferramentas para computar características de desempenho previsto; e 4) Apresentação de resultados de desempenho, computados por ferramentas de modelagem ou obtidos por meio de testes.

3.1 – Expressando QoS através da UML-RT

Com a utilização do *framework* para análise de desempenho disponível no perfil UML-RT, os requisitos de QoS podem ser claramente expressos através de uma série de

estereótipos. Estes estereótipos permitem uma análise efetiva da influência de tais requisitos no projeto da aplicação [AAGEDAL, ECKLUND 2002].

Nos modelos do perfil UML-RT, um recurso é visto como um servidor que provê um ou mais serviços para um cliente. A limitação física do recurso é representada através dos seus atributos de QoS. Em geral, os atributos de QoS caracterizam *quão bem* um recurso realiza um serviço. O modelo das limitações de um recurso é feito através dos estereótipos *PAresource* e *PAhost.*

O conceito de QoS é bastante amplo, podendo ser usado para modelar diferentes propriedades. Este trabalho procurou seguir o conceito apresentado em [VOGEL 1995], segundo o qual as características modeladas são preferencialmente quantitativas (tamanho, tempo de serviço, capacidade, tempo de resposta, latência, largura de banda, *jitter*), mas também podem ser especificações qualitativas (compartilhamento, política de escalonamento ou modelo de falha), que no final vão provocar algum impacto quantitativo. Estas informações aparecem de forma clara nos modelos UML-RT com o uso dos estereótipos *PAresource* e *PAhost.* Já informações que tratam de aspectos quantitativos de execução de ações são representadas pelo estereótipo *PAstep*.

Naturalmente, não é suficiente conhecer a QoS que um recurso pode oferecer. Também é necessário saber o que é solicitado pelo cliente. Isto sugere a diferenciação entre QoS oferecida (do lado do recurso) e QoS requerida (do lado do cliente). Na análise quantitativa o que se busca determinar é a compatibilidade de QoS destes dois extremos. Com isto, o que se procura entender é como se dá a interação existente entre os dois extremos do sistema e que fatores a influenciam. O foco é, portanto, o contexto da interação. Os estereótipos *PAcontext*, *PAclosedLoad* e *PAopenLoad* fornecem a semântica desejada para estudo e análise deste aspecto do projeto.

4 – Estudo de Caso

O estudo de caso apresentado neste artigo é o de uma aplicação de vídeo-conferência. Um sistema como este apresenta uma série de características rigorosas no atendimento de limites temporais que merecem especial atenção durante o seu projeto. Caso estes requisitos não sejam devidamente expressos, sua implementação poderá não atender às necessidades para as quais foi projetado [BOJKOVIC, MILOVANOVIC 2002]. Vários requisitos de QoS poderiam ser tratados neste estudo de caso, porém, restringiu-se o escopo deste trabalho à análise de pelo menos dois, latência e *jitter*. A seguir serão apresentados o problema e os modelos utilizados nesta análise.

4.1 – Descrição do Sistema

Uma descrição básica da funcionalidade do sistema pode ser dada nos seguintes termos: Tem-se duas ou mais estações de trabalho que desejam se comunicar enviando imagem e som para a(s) outra(s) estação(ões) via uma aplicação *web*. Por razões de simplicidade, será considerado somente o sub-caso no qual existem apenas duas estações no sistema, porém a adição de outras estações em nada dificulta a análise ou o projeto. Dado que a comunicação e a troca de mensagens são sempre ponto-a-ponto, o que limitará o número de estações constituintes do sistema será a capacidade de cada uma em atender solicitações de outras e também de receber e tratar pacotes dos outros nós da rede. Portanto, o sistema consiste numa distribuição *peer-to-peer*, na qual cada nó é ao mesmo tempo servidor e cliente. Como servidor cada nó recebe uma requisição de um cliente e aceitando tal requisição, passa a transmitir pacotes de vídeo para esta estação. Como cliente, uma estação realiza uma requisição acessando uma determinada

URL e aguarda sua aceitação. Sendo aceita, a estação cliente passa a receber pacotes de vídeo que devem ser reproduzidos localmente.

Após a descrição básica do sistema, serão apresentados os requisitos de desempenho que devem ser atendidos. Os seguintes requisitos de desempenho serão analisados:

1 - o retardo máximo (latência) admitido entre o pedido de conexão e a sua aceitação e confirmação não deve exceder 500 milissegundos em 98% dos casos. Logo a probabilidade do retardo ser maior que 500 milissegundos deve ser menor que 2%.

2 - a apresentação dos *frames* ao usuário deve seguir um intervalo regular de 30 milissegundos, sendo garantido que este intervalo deve ser cumprido sem atraso em 99% dos casos. Isto significa que a probabilidade do intervalo entre *frames* ser superior a 30 milissegundos (*jitter*) deve ser menor que 1%.

Para que seja possível realizar a análise sobre os requisitos apresentados, faz-se necessário o conhecimento de características de execução dos componentes do sistema. Algumas destas características, estimadas com base em dados apresentados em [BOJKOVIC, MILOVANOVIC 2002] e [HALSALL 2000], são listadas a seguir:

1 – tempo médio de processamento de um *frame* no servidor (sem contar o tempo de captura da imagem e do som): 10 milissegundos;

2 – número de pacotes necessários para enviar um *frame* pela rede: 50

3 – latência média da rede: 3 milissegundos;

4 – tempo médio de processamento de um *frame* no cliente: 15 milissegundos;

5 – o número de *frames* por conferência é uma variável do sistema: $N;

6 – o número de usuários por conferência é uma variável do sistema: $NUsers.

4.2 – Modelos UML-RT

Com base na descrição do sistema e seus requisitos, é possível a criação de diagramas que facilitam a análise do mesmo, e com isto ponderar sobre as possíveis alternativas de projeto. Os diagramas apresentados neste trabalho foram gerados pela ferramenta Real-Time Studio da Artisan [ARTISAN 2004].

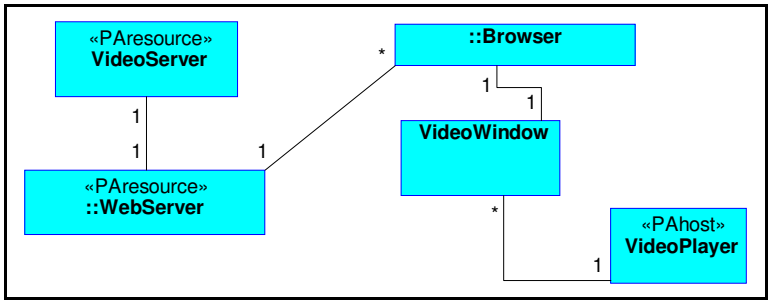

Figure 1. Diagrama de Classes

Na figura 1 observa-se uma visão parcial do diagrama de classes para a aplicação de vídeo-conferência. O diagrama de classes simplificado não apresenta os atributos nem as operações de cada classe. Optou-se por esta visualização para melhor destacar a utilização dos estereótipos no modelo. Pela observação do diagrama, a aplicação constitui-se, no lado cliente, basicamente de um navegador (Browser) pelo qual o usuário entra no sistema, um visualizador (VideoWindow) e um reprodutor de vídeo (VideoPlayer). Este último, por ser um recurso que pode ser compartilhado por mais de um visualizador, foi estereotipado como um *PAhost*. As classes que

representam a parte do servidor foram estereotipadas com *PAresource*, uma vez que apresentam características como capacidade de processamento, tempo de resposta e vazão da rede (*throughput*), todas relevantes para a análise do sistema.

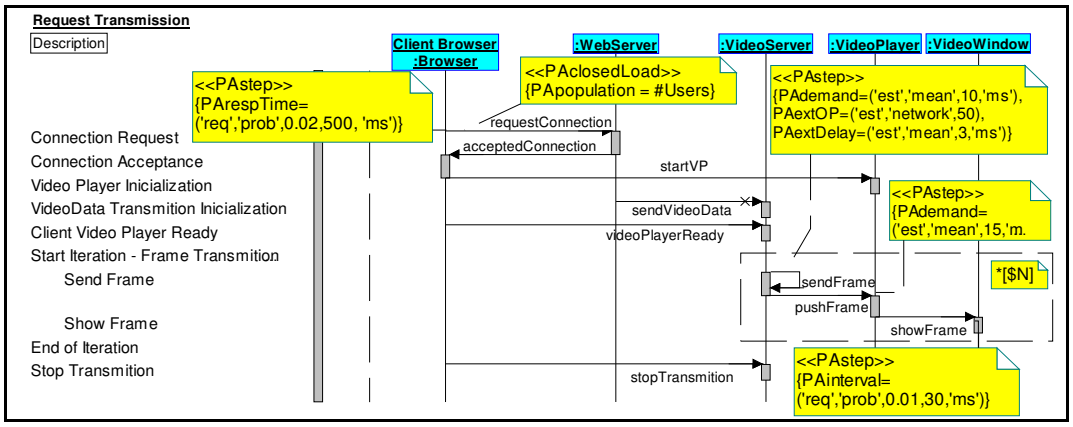

Figure 2. Diagrama de Seqüência

A partir do diagrama de classes e da descrição apresentada, escolheu-se o diagrama de seqüência para realização da análise dos requisitos temporais da aplicação. Esta escolha foi feita porque os objetivos principais da análise são a dinâmica de interação entre os componentes do sistema numa seqüência temporal, e o relacionamento entre os requisitos e características do sistema neste contexto. O diagrama de seqüência é a ferramenta mais adequada para esta tarefa.

Na figura 2, observa-se que os dois requisitos temporais descritos na sub-seção anterior encontram-se representados pelas *tags* (valores etiquetados) *PArespTime* (requisito do tempo de resposta para conexão), associada à chamada da função *requestConnection*, e *PAinterval* (requisito de intervalo entre *frames*), associada à execução da função *showFrame*. Na descrição destas *tags* encontram-se as abreviaturas "req" e "prob" que significam respectivamente "requisito" e "probabilidade", além da unidade de tempo milissegundos, abreviada para "ms".

As características dos componentes do sistema (tempo de processamento) encontram-se expressas através da *tag PAdemand*, enquanto as características de interação e contexto são expressas através das *tags PAextOP*, *PAdelay* e *PApopulation*. Na sua descrição encontram-se as abreviaturas "est" e "mean" que significam respectivamente "valor estimado" e "média", além dos valores nominais característicos. Cada *tag* está associada a um momento da execução do cenário. Por exemplo, quando ocorre a chamada da função *sendFrame*, os fatores relevantes a serem analisados dizem respeito: 1) ao tempo de processamento daquele *frame* que será transmitido; 2) ao número de pacotes necessários para enviar um *frame*; e 3) ao retardo imposto pela rede. Estas informações estão descritas na *tag* ligada à seta que representa a chamada da função *sendFrame*.

Como se tem uma iteração das chamadas *sendFrame*, *pushFrame* e *showFrame* igual ao número de vezes quantos forem os *frames* a serem enviados (variável definida na descrição apresentada na sub-seção 4.1), destacou-se estas chamadas com um retângulo pontilhado. Isto destaca a representação da ocorrência das $N execuções.

Deve ser ressaltada a expressividade do modelo apresentado na figura 2, uma vez que nele se encontram todas as informações necessárias para a análise dos dois requisitos de desempenho propostos.

5 – Conclusão

Este trabalho buscou chamar atenção para a possibilidade do uso da UML-RT na modelagem de sistemas interativos baseados em aplicações de rede. Como estes sistemas apresentam rigorosos requisitos de QoS, com prazos rígidos a serem cumpridos, tais como em sistemas de controle industrial, esta proposta mostra-se bastante pertinente.

Surge assim uma nova perspectiva para a utilização dos recursos providos pela UML-RT. Esta linguagem poderia ser utilizada para formalização de requisitos de QoS em contratos de serviços entre operadoras de telecomunicação e entre operadora e usuário. Desta forma, as partes contratantes ficariam obrigadas a cumprir exigências formalmente descritas segundo um modelo padronizado.

O estudo de caso apresentado permitiu visualizar o poder de expressão da extensão tempo-real da UML bem como sua adequação para modelagem de sistemas interativos baseados em rede. Deve-se considerar que atualmente a utilização de UML-RT neste tipo de sistema se mostra tão adequada quanto em sistemas que necessitam de concepção paralela de hardware e software. Segundo [SELIC 1999], o uso de UML no projeto destes sistemas também não era algo tão evidente na época em que foi proposto.

6 – Referências Bibliográficas

[AAGEDAL, ECKLUND 2002] Aagedal, J. O., Ecklund, E. F "Modelling QoS: Towards a UML Profile". In: <<UML>> 2002, Dresden, Germany, September 20 – Oct 4, 2002. Published in Spring LNCS 2460, ISBN 3-540-44254-5, pp. 275-289.

[ARTISAN 2004] ARTISAN Software. Real-Time Studio. http://www.artisansw.com/.

[BOJKOVIC, MILOVANOVIC 2002] Bojkovic, Z. S.; Milovanovic, D. A. Multimedia Comunication Systems: Techniques, Standards, and Networks, Prantice Hall, 2002.

[DOUGLASS 1999] Douglass, Bruce P. Real-Time UML Second Edition – Developing Efficient Objects for Embedded Systems - Addison-Wesley, 1999.

[EDWARDS 1997] Edwards, S., Lavagno, L., Lee, A., Sangiovanni-Vincentelli, A. "Design of Embedded Systems: Formal Models, Validation, and Synthesis", In Proc. IEEE, vol. 85, pp. 366-390, March 1997.

[HALSALL 2000] Halsall, F. Multimedia Communications: Applications, Networks, Protocols, and Standards, Addison-Wesley Publishing, 2000.

[OMG 2004] OBJECT MANAGEMENT GROUP. UML Profile for Schedulability, Performance, and Time Specification, Feb 2004. <http://www.omg.org/cgi-bin/doc?ptc/04-02-01>.

[SHAW 2001] Shaw, A. Real-Time Systems and Software, John Wiley& Sons, 2001.

[SELIC 1999] Selic, B. "Turning Clockwise: Using UML in the Real Time Domain", Communications of the ACM, v. 42, n.10: 46-54, Oct 1999.

[TSAI, YANG, CHANG 1995] Tsai, J. P.,Yang, S. J., Chang, Y. "Timing Constraint Petri Nets and Their Application to Schedulability Analysis of Real-Time System Specifications", IEEE Transactions On Software Engineering. Vol.21, N. I, Jan 1995.

[VOGEL 1995] Vogel, L. A. et al. "Distributed Multimedia and QoS: A Survey", IEEE Multimedia, Summer 1995.