UMA METODOLOGIA DE APOIO À ANÁLISE E AO DESENVOLVIMENTO DE MODELOS PARA SIMULAÇÃO DE PLATAFORMAS DE DISTRIBUIÇÃO

Adriano Augusto de Souza
Escola Técnica da Federal da Paraíba
adriano@jpa.etfpb.br
Paulo Roberto Freire Cunha
Universidade Federal Pernambuco
prfc@di.ufpe.br

Resumo

Mesmo dispondo de ferramentas eficientes para a construção de plataformas de distribuição, engenheiros de software necessitam, frequentemente, testar novos mecanismos e extensões que melhorem o desempenho durante a execução das plataformas de distribuição em diversos tipos de cenários. As melhores soluções disponíveis atualmente são a implementação direta da proposta, a criação de protótipos para testes ou a construção de modelos de simulação baseados em teoria das filas, gerando, devido a demora e a complexidade para se obter a melhor estratégia, um grande desperdício de tempo e de recursos, como também prejuízos no controle da qualidade do software produzido. Nesse artigo, apresentamos uma metodologia de apoio, denominada DMSPD (Desenvolvimento de Modelos para Simulação de Plataformas de Distribuição), que tem como objetivo facilitar a análise e o desenvolvimento de modelos para simulação de plataformas de distribuição.

Palavras Chaves: Engenharia de Software, Middleware, Sistemas Distribuídos, Especificação Formal.

1 Introdução

A partir da difusão das redes de computadores, ocasionada pela necessidade do compartilhamento de recursos entre os diversos sistemas computacionais, os usuários sentiram necessidade de uma maior integração e trabalho cooperativo entre eles. Dessa forma, ambientes de processamento distribuído de acesso ilimitado têm-se tornado imprescindíveis. Tais ambientes tendem a oferecer e viabilizar o acesso às informações, seja ele local ou remoto, de uma forma transparente e homogênea para o usuário [JUS88].

Entretanto, para que tais ambientes consigam ter o sucesso esperado, os problemas decorrentes da heterogeneidade do processamento distribuído precisam ser resolvidos. Metodologias, ferramentas e tecnologias para o desenvolvimento desse tipo de ambiente estão sendo desenvolvidas e aprimoradas, surgindo com isso as plataformas de distribuição ou *middleware*, que visam criar uma infra-estrutura básica para o desenvolvimento de sistemas distribuídos. Elas se propõem a estabelecer a interconectividade entre os recursos através de várias funções de distribuição das aplicações, como notificação de eventos, suporte à transação, segurança e nomeação; além de prover transparências de distribuição, como acesso, localização, migração, replicação, persistência e falhas [Far95].

Embora a idéia das plataformas de distribuição seja muito interessante, colocá-las em prática é uma atividade bastante complexa. Na proporção em que as redes tornam-se maiores (em extensão) e heterogêneas (formadas por plataformas de hardware e *software* distintas), esta atividade torna-se mais complexa ainda. Consequentemente, sua construção não pode ser realizada somente pelo esforço humano [Ram95]. Logo, modelos para simulação estão sendo desenvolvidos como forma de podermos exercitar experimentalmente os mecanismos propostos às plataformas antes mesmo de implementá-las. Neste sentido, tais experimentos permitem realizarmos inferências sem construir os mecanismos, pois estes são apenas sistemas propostos; sem pertubá-los, já que são sistemas de alto custo operacional ou sem segurança para que se façam experimentos; e sem destruí-los, já que o objetivo da experimentação é determinar os limites de funcionamento do sistema.

Todavia, os modelos para simulação de plataformas de distribuição, por se basearem normalmente em teoria das filas, têm-se apresentado bastante complexos, gerando dificuldades na produção das inferências. Desse modo, outras formas de desenvolvimentos de modelos para a simulação estão sendo estudadas, utilizadas e melhoradas [Mit94]. No entanto, um item importante como o desenvolvimento de simulações baseadas no paradigma de orientação a objetos não foi totalmente explorado. Por isso, na seção seguinte, apresentamos uma metodologia de apoio a este tipo de atividade, denominada DMSPD (Desenvolvimento de Modelos para Simulação de Plataformas de Distribuição), que tem como objetivo facilitar a análise e o desenvolvimento de simulações para as plataformas de distribuição, tendo como base o paradigma de orientação a objetos. Na seção 3, exemplificamos a utilização da abordagem DMSPD e, na seção 4, apresentamos algumas conclusões e trabalhos futuros.

2 A Metodologia de Apoio DMSPD

A análise matemática realizada por um experimento de simulação em uma plataforma de distribuição representa uma abordagem que pode ser usada para entender o desempenho projetado. A metodologia de apoio DMSPD congrega um conjunto de atividades que facilita a análise e o desenvolvimento das plataformas de distribuição. Estruturada em fases, a metodologia de apoio DMSPD especifica, de forma disciplinada e baseada nas técnicas de engenharia de *software* atuais, que atividades devem ser realizadas para o desenvolvimento das plataformas de distribuição. A seguir, cada uma das fases do procedimento DMSPD é descrita.

2.1 Fase 1 - Especificação do Modelo

Em ciência da computação, modelo é uma abstração do sistema, que é uma coleção de itens, entre os quais se possa encontrar ou definir alguma relação, que são objeto de estudo ou interesse [Soa90]. A construção de modelos é um processo muito complexo e, em muitos campos, uma arte. Ao criar um modelo, uma das tarefas mais difíceis que enfrenta o modelador é a decisão sobre que elementos dos sistema devem ser incluídos no modelo. Para tomar esta decisão, o modelador precisa antes de tudo estabelecer o propósito do modelo.

Desse modo, como nosso propósito neste artigo é apresentar uma abordagem que permita entender o desempenho projetado através de simulações, pois tais estruturas permitem o exercício do modelo como forma de mimetizar o comportamento do sistema, necessitamos de uma técnica que permita refletir de fato a compreensão do comportamento do sistema em seu ambiente ao longo do tempo, mas que não produza uma complexidade que torne o resultado da simulação um problema de difícil solução, como os modelos resultantes das redes de filas, fator bastante comum em modelos para simulação de plataformas de distribuição. Desse modo, a técnica usada será a metodologia OMT (*Object Modelling Technique*) [Rum94].

Tal metodologia compõe-se de três pontos de vista diferentes chamados de modelo de objetos (que descreve a estrutura estática de um sistema em termos de objetos e relacionamentos correspondentes a entidades do mundo real), modelo dinâmico (que descreve a estrutura de controle de um sistema em termos de eventos e de estados) e modelo funcional (que descreve a estrutura computacional de um sistema em termos de valores e de funções). Como cada ponto de vista contém referências aos outros, julgamo-la como a mais adequada para representação da especificação de um modelo para simulação, pois ela permite que se construa um modelo abrangente que seja suficientemente preciso para ser confiável e claro o bastante para ser útil, possuindo ferramentas que proporcionam extensos mecanismos para inspecionar e depurar a especificação e para recuperar informações.

2.2 Fase 2 - Garantindo a Qualidade do Modelo

Para que se possa realizar adequadamente a garantia da qualidade do modelo apresentado, dados sobre o processo de engenharia de software devem ser compilados, avaliados e divulgados. Entre as métricas de qualidade de software, incluem-se tanto as medidas diretas como as indiretas do *software* [Pre95].

Quando métodos formais são usados durante o desenvolvimento de um projeto de software, eles servem de base para a verificação dos programas e, por conseguinte, possibilitam que o engenheiro de software descubra e corrija erros que, de outra forma, poderiam passar despercebidos [Pre95]. Utilizamos como abordagem formal para garantia da qualidade do modelo especificado na fase 1 a linguagem de especificação LOTOS (*Language Of Temporal Ordering Specification*). Ela foi escolhida devido a três critérios. Primeiro, ela é uma linguagem algébrica que foi desenvolvida pelos especialistas em Técnicas de Descrição Formal da ISO (*International Organization of Standardization*), o que facilita a prova de corretude. Segundo, esta linguagem é uma das técnicas formais para a especificação e análise de sistemas distribuídos abertos. A sua definição foi motivada pela necessidade de se descrever os serviços e protocolos dos diversos níveis da arquitetura de sistemas abertos OSI (*Open Systems Interconnection*) [Pal96]. Por último, apesar de ter a descrição de sistemas abertos como o seu objetivo principal, LOTOS é uma técnica muito poderosa que pode ser utilizada para a descrição de sistemas distribuídos em geral [Que94].

Pode-se argumentar que o programa de computador é um objeto matemático [Som89]. Dessa maneira, para que o modelo OMT seja considerado um objeto matemático, facilitando a prova de corretude, teremos que descrever uma semântica formal para ele ou mapeá-lo numa linguagem de especificação formal como LOTOS. Entendemos que a segunda alternativa será mais válida, devido a ela possibilitar a demonstração da correspondência entre a especificação formal e o modelo. Assim, como a linguagem LOTOS tem como ênfase principal expressar o comportamento dos objetos por meio de uma hierarquia de definições de processos, consideramos que o mapeamento do modelo dinâmico OMT em LOTOS será um caminho mais natural. Dessa forma, tomaremos como referência o mapeamento estabelecido pela Tabela 1, a qual foi planejada por nós, tendo como base a metodologia desenvolvida em [Ros96], a qual implementa projetos formais LOTOS na plataforma de distribuição ANSAware.

OMT	*LOTOS*
Classe	Processo de mais alto nível
Estado	Sub-processo
Evento	Ação
Estado Seguinte	Instanciação de Processo

Tabela 1 - Mapeamento OMT em LOTOS

2.3 Fase 3 - Simulação

Ao desenvolver um modelo para simulação, um engenheiro de software precisa selecionar a estrutura conceitual, na qual o modelo vai se apoiar, para a descrição do sistema. A estrutura conterá o enfoque (ou visão) dentro do qual as relações funcionais entre os elementos do sistema são percebidas e descritas [Soa90].

O objetivo de uma simulação é reproduzir as atividades engajadas e, a partir daí, conhecer algo sobre o comportamento e o desempenho do sistema. A forma mais natural de se conseguir isto, é através de uma simulação orientada a eventos, onde um sistema é modelado pela definição das mudanças que ocorrem no tempo de evento, ficando como tarefa do engenheiro de software determinar os eventos que podem causar a mudança no estado do sistema e então desenvolver a lógica associada a cada tipo de evento, pois a simulação será produzida pela execução da lógica associada a cada evento, em uma sequência ordenada de tempo.

Dessa forma, nessa fase, baseamos nosso trabalho no tipo de simulação orientada a eventos, onde usamos como guia principal da implementação da simulação a descrição do modelo dinâmico produzido pela especificação OMT na fase 1, por considerarmos que este modelo permite um mapeamento bastante direto entre o modelo OMT e a ferramenta de simulação orientada a eventos a ser utilizada. Além disso, sabemos também que fazendo isso, não estamos indo de encontro à metodologia

OMT, pois a mesma pode ser implementada com a utilização de linguagens baseadas em objetos ou não, desde que se mantenha disciplina em relação ao paradigma orientado a objetos.

3 Utilização da Metodologia de Apoio DMSPD

Nesta seção, exemplificamos a utilização da metodologia de apoio DMSPD através de uma extensão da plataforma de distribuição ANSAware para suportar a computação móvel [Sou96].

3.1 M-REX

ANSAware é uma plataforma de distribuição, desenvolvida em conformidade com o RM-ODP (*Reference Model - Open Distributed Processing*), com o objetivo de apoiar o desenvolvimento e a execução de aplicações distribuídas [Ros96]. Seu protocolo de comunicação de dados, REX (*Remote Execution protocol*), é baseado no mecanismo de RPC (*Remote Procedure Call*). Tal mecanismo permite a comunicação entre diferentes partes de aplicações distribuídas, uma vez, que fica especificado nos *stubs* como será efetuada a transferência de informações entre as duas partes que compõem a aplicação.

O protocolo REX trabalha bem quando o ambiente da rede é estático. Entretanto, como ele não foi projetado para trabalhar em ambientes dinâmicos, como é o caso do ambiente de computação móvel, sua abordagem se torna pouco produtiva [Kir95]. Neste caso, a fim de prover transparência para suportar uma computação móvel mais eficiente foi proposto ao ANSAware, em [Sou96], a criação do protocolo M-REX (*Mobile - Remote EXecution protocol*). Tal protocolo possui mecanismos que corrigem a perda de desempenho, oferecendo suporte as desconexões e aos *handoffs*, fatores bastantes comuns no ambiente de computação móvel. Dentre os mecanismos propostos encontram-se a utilização de ligações dinâmicas que possibilitam a criação de um balanceamento de carga controlado pelo Trader (servidor de nomes) entre as ligações cliente-servidor, o uso de um protocolo de transporte confiável para o ambiente de computação móvel, um controle apropriado de retransmissão de pedidos como forma de conservação de energia do *host* móvel, um suporte à operação de desconexão possibilitando que um *host* móvel deixe alguma atividade sendo realizada enquanto ele se desconecta, uma monitoração da qualidade do serviço e a implementação de serviços de segurança e contabilização dos custos como forma de garantir um uso razoável e ao mesmo tempo eficiente dos recursos da plataforma.

3.1.1 Fase 1 - Especificação do Modelo

Para criação da especificação OMT (modelo de objetos, dinâmico e funcional) utilizamos a ferramenta CASE (*Computer - Aided Software Engineering*) de análise e projeto **Paradigm Plus v3.0** [Pro95]. Tal ferramenta possibilita ao engenheiro de software auxílio na criação da descrição do sistema e também na avaliação da qualidade da especificação produzida. Ao executar a verificação da consistência e a validade do modelo, a ferramenta Paradigm proporciona ao engenheiro de software certo grau de esclarecimento sobre a representação da análise e ajuda a eliminar erros antes que eles se propaguem pela outras etapas do desenvolvimento do software.

O modelo de objetos resultante é apresentado na Figura 1. Observe que o *Trader* foi modelado conforme uma associação como classe. Nesse caso as ligações poderiam estar sujeitas às operações. Outro detalhe, é que presumimos que os processos servidores no M-REX só seriam executados em *hosts* fixos. Caso eles pudessem ser executados também em *hosts* móveis, o modelo poderia ser expandido para manipular essa situação. Entretanto, admitimos que os clientes estariam satisfeitos com essa limitação, em virtude de não fazer sentido um processo servidor ser executado num equipamento que tenha capacidade de energia reduzida.

Um resumo do modelo dinâmico resultante é exibido na Figura 2. O cenário tomado como base consiste num serviço orientado a conexão, onde o cliente solicita ao *Trader* a melhor localização de tal serviço, para só depois efetuar os pedidos. Os objetos ***host* móvel**, **estação rádio base**, ***Trader*** e ***host* fixo** são atores que trocam eventos entre si. Os objetos **área de sombreamento** e **área de cobertura** são objetos passivos que são manipulados e não

intercambiam eventos. Os objetos estação rádio base e *Trader* juntos logicamente formam o agente M-REX localizado no MSC (*Mobile Switching Center*), logo serão tratados por nós a nível de modelo dinâmico como um só.

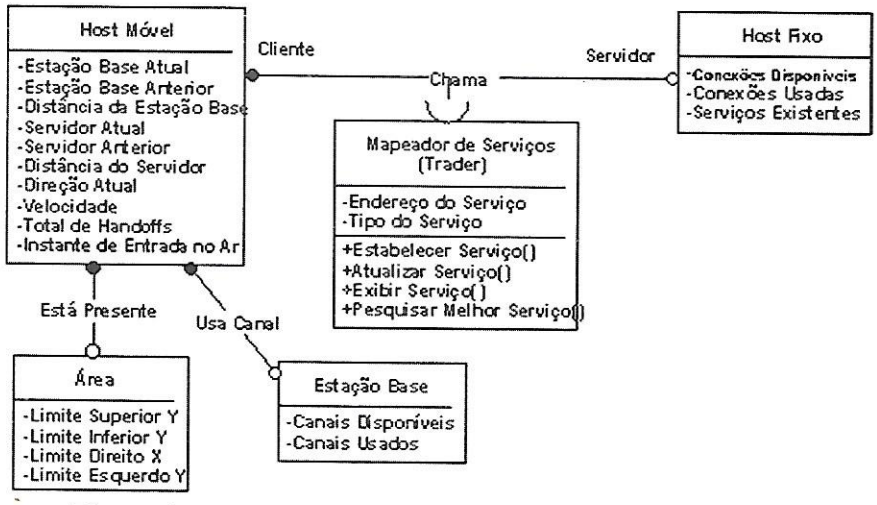


Figura 1 - Resumo do Modelo de Objetos para o M-REX

3.1.2 Fase 2 - Garantindo a Qualidade do Modelo

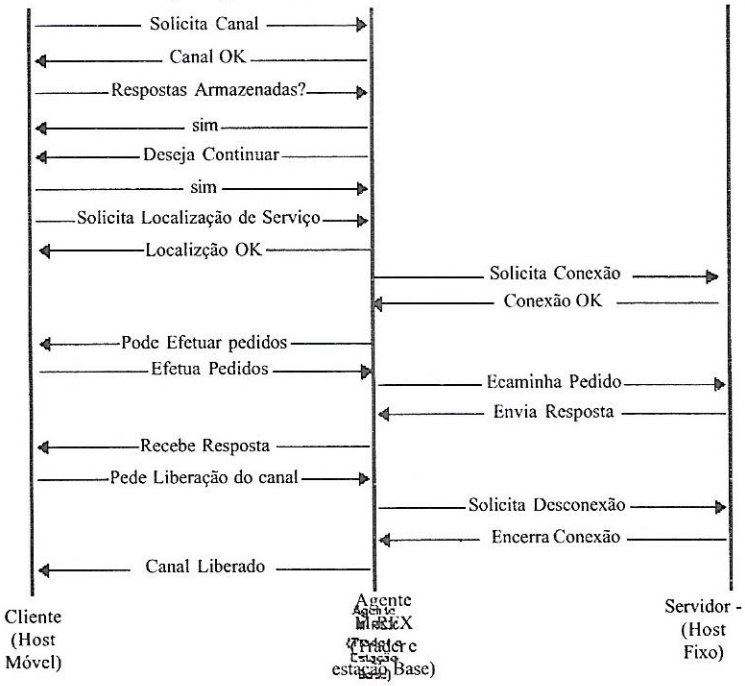
Para auxiliar na escrita, leitura, verificação e validação da especificação da fase anterior utilizamos a ferramenta de especificação LITE (*LOTOS Integrated Tool Environment*) [Que94]. Tal ferramenta foi proposta pelo projeto LOTOSPHERE com o objetivo de criar um ambiente de desenvolvimento de sistemas distribuídos que suporte todas as fases da vida de um sistema [Que94]. Como resultado tivemos a seguinte especificação:

```
SPECIFICATION ProcessodeHandoff [CanaldeControle,CanaldeDados]:NOEXIT
TYPE Status is Boolean
    sorts Status
    opns
        EntrouemSombreamento: -> Status
        SaiudaÁreadeAbrangência: -> Status
        Desligou: -> Status
        Handoff: -> Status
        SaiudoSombreamento: -> Status
        TempoDemais: -> Status
        Saiu: -> Status
        HandoffEfetuado: -> Status
        CanalnaoDisponível: -> Status
ENDTYPE
BEHAVIOUR
    HostMovel [CanaldeControle,CanaldeDados]
        |[CanaldeControle]|
    (Agente [CanaldeControle]
        |[CanaldeControle]|
    HostFixo[CanaldeControle,CanaldeDados])
WHERE
PROCESS HostMovel[CanaldeControle,CanaldeDados]:NOEXIT
:=
ComunicacãoAtiva [CanaldeControle,CanaldeDados]
[>
OcorrênciaExterna [CanaldeControle,CanaldeDados]
WHERE
    PROCESS OcorrênciaExtena[CanaldeControle,CanaldeDados]:NOEXIT
        :=
    CanaldeControle!EntrouemSombreamento;
            Sombreamento[CanaldeControle,CanaldeDados]
    []
    CanaldeControle!SaiudaÁreadeAbrangência;
        Desconectado[CanaldeControle,CanaldeDados]
    []
    CanaldeControle!Desligou-se;
        Desconectado[CanaldeControle,CanaldeDados]
```

```
[]
CanaldeControle!Handoff;
    TratandoHandoff[CanaldeControle,CanaldeDados]
    WHERE
    PROCESS Sombreamento[CanaldeControle,CanaldeDados]:NOEXIT
    :=
    CanaldeControle!SaiudoSombreamento;
        HostMovel[CanaldeControle,CanaldeDados]
    []
    CanaldeControle!TempoDemais;
        Desconectado[CanaldeControle,CanaldeDados]
    []
    CanaldeControle!Desligou-se;
        Desconectado[CanaldeControle,CanaldeDados]
    ENDPROC
    PROCESS Desconectado[CanaldeControle,CanaldeDados]:NOEXIT
    :=
    CanaldeControle!Saiu;
        Desconectado[CanaldeControle,CanaldeDados]
    ENDPROC
    PROCESS TratandoHandoff[CanaldeControle,CanaldeDados]:NOEXIT
    :=
    CanaldeControle!HandoffEfetuado;
        HostMovel[CanaldeControle,CanaldeDados]
    []
    CanaldeControle!CanalnaoDisponivel;
        Desconectado[CanaldeControle,CanaldeDados]
    []
    CanaldeControle!Desligou;
        Desconectado[CanaldeControle,CanaldeDados]
    ENDPROC
  ENDPROC
ENDPROC
ENDSPEC
```

A especificação inicial do M-REX, em nível mais abstrato, representa a composição paralela dos processos **HostMovel**, **Agente** e **HostFixo**. Estes processos representam respectivamente todos os clientes, agentes e servidores do modelo proposto M-REX. As estações base, o centros de comutação, os MSRs (*Mobility Support Routers*) são agregados no processo **Agente**, refletindo a estrutura física de um sistema celular.

As portas **CanaldeControle** e **CanaldeDados** são usadas para representar as informações de controle e de dados (ou voz) respectivamente. Note que somente a porta **CanaldeDados** não é escondida do sistema. Isto acontece porque a mobilidade dos *hosts* móveis são descritas como eventos externos ao sistema.



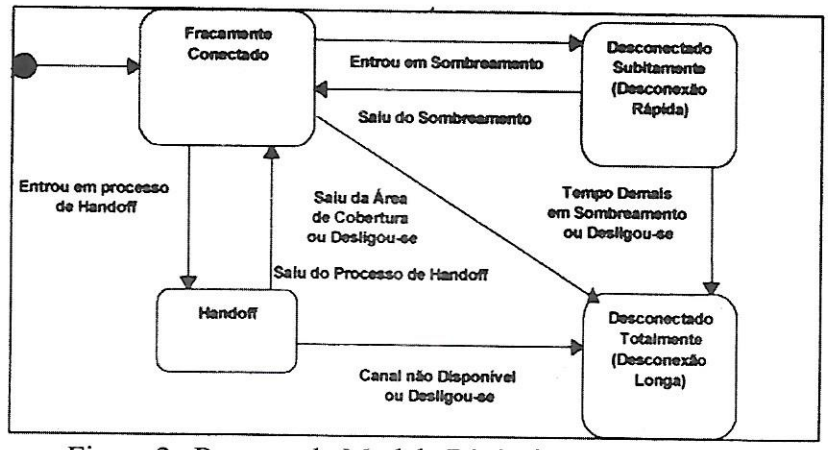


Figura 2 - Resumo do Modelo Dinâmico para o M-REX

Seguindo o refinamento passo-a-passo, definimos o processo **HostMovel**. Tal processo tem como operador a expressão [>, que representa a preempção de processo, ou seja, a modelagem de comportamento do processo **Comunicação Ativa** pode ser preemptado pelo processo **Ocorrência Externa**.

Por fim, na especificação acima percebe-se que apresentamos uma especificação formal apenas para o diagrama de estados do *host* móvel. Tal ação foi realizada em virtude de considerarmos que apenas esse diagrama, devido à complexidade provocada pelo *handoff* (passagem de controle da ligação de uma estação rádio base para outra estação rádio base), necessitava ser validada através de uma abordagem formal. O mesmo não acontecendo com os outros diagramas, que podem ser omitidos, devido à pouca complexidade associada aos mesmos.

3.1.3 Fase 3 - Simulação

O objetivo real de simular a extensão M-REX da plataforma de distribuição ANSAware é verificar se ela cumprirá de fato as expectativas quando estiver em funcionamento no mundo real. Para isso, utilizamos uma ferramenta que permite criarmos e executarmos cenários baseados no modelo proposto na seção 3.1.1. Tal ferramenta será o **BONeS Designer** [Alt96a].

O BONeS Designer (*Block Oriented Network Simulation Designer*) consiste de um pacote de software para modelagem e simulação de sistemas orientados a evento. Ele tem como componentes básicos os seguintes produtos: *Block Diagram Editor*, *Data Structure Editor*, *Simulation Manager*, *Post Processor* e *Symbol Editor*. O *Block Diagram Editor* cria, edita, documenta e armazena diagramas de blocos, tendo como característica fundamental o uso da hierarquia, composição e reusabilidade dos blocos, além de possuir uma vasta biblioteca de blocos já implementados. O *Data Sructure Editor* cria, edita, documenta e armazena estruturas de dados, tendo como aspecto básico o uso da hierarquia e da herança de tipos de dados. O *Simulation Manager* gera, executa e monitora programas de simulação, possuindo ainda a capacidade de produzir animações. O *Post Processor* analisa e exibe resultados da simulação. Enquanto o *Symbol Editor* cria símbolos descritivos para diagramas de blocos.

A implementação da simulação do M-REX pôde aproveitar muitas das funções da simulação do sistema de telefonia celular móvel existente no BONeS Designer [Alt96b]. As operações de inicialização foram as seguintes: *Init*, *Create & Save All The Base Stations*, *Create & Save All The Servers*, *Create Coverage Area* e *Create Mobile Users*. Essas operações servem para determinar a localização das estações rádio base e dos servidores, além de identificar a área de cobertura total, como também as zonas de sombreamento.

Os eventos foram sequenciados conforme a dinâmica do protocolo. O escalonamento é feito de acordo com a presença de pacotes de controle ou de dados na subrede. Para o M-REX as funções abaixo são engatilhadas: *Assign Inicial Server To Mobile Users*, *Assign Inicial Connection To Mobile Users*, *Assign Base Station To Mobile Users*, *Assign Channel To Mobile Users*, *Data*

Delay Mobile Users, *Deleted Expired Mobile Users*, *Move Mobile Users*, *Channel Quality Ok?*, *Retransmission Data Delay*, *Server Quality Ok? (Distance)*, *Retransmission Data Delay - Server Changes*, *Release Connection*, *Assign Server To Mobile Users*, *Assign Connection To Mobile Users*, *Release Channel*, *Mobile User is Blocked*, *Mobile User Out of Coverage Area*, *Mobile User Session is Over* e *Stat Collect & Compute*.

Quando o cliente parte do programa, o bloco *Stat Collect & Compute* coleta e calcula as estatísticas já pré-definidas. Já os blocos *Mobile User is Blocked*, *Mobile User Out of Coverage*, *Area Mobile User Session is Over* são para criar efeitos visuais numa possível animação da simulação.

4 Considerações Finais

A metodologia de apoio DMSPD é resultado de uma reflexão decorrente da nossa experiência em avaliação de desempenho de plataformas de distribuição e de protocolos de redes de computadores. Embora genérica, ela pode ser usada como uma carta de navegação para análise e desenvolvimento de modelos para simulação de plataformas de distribuição, pois viabiliza a construção de modelos mais simples e fáceis de manipular do que as redes de filas produzidas pela teoria da filas. Uma outra vantagem é que a abordagem proposta poderá ser aplicável também a outros tipos de simulações, desde que estas necessitem da criação de um modelo de simulação mais complexo para representação do comportamento do sistema, como é o caso dos modelos utilizados em plataformas de distribuição, em vários protocolos de redes de computadores e em um grande número de sistemas de tempo real.

Na abordagem DMSPD, enfatizamos a análise para o desenvolvimento de modelos para simulação de plataformas de distribuição e não a análise de dados obtidos de elementos simulados. Logo, um ponto importante com respeito a trabalhos futuros consistirá no estudo de viabilidade para construção de uma ferramenta que permita, pelo menos em parte, automatizar o processo da abordagem, ou seja, passar de um modelo dinâmico OMT e de uma especificação formal LOTOS para o projeto de simulação do BONeS Designer.

Por fim, o uso da abordagem DMSPD possibilitará que os desenvolvedores de plataformas de distribuição pratiquem as propostas de mecanismos de suporte às plataformas interativamente. Com isso, eles terão condições de avaliação de desempenho sob condições aleatórias, tanto em situações típicas como atípicas, com grande economia de tempo e sem necessidade de gastos com custos inerentes a implementações.

5 Bibliografia

[Alt96a] Alta Group of Cadence Design Systems. BONeS Designer Release 3.6. Modelling Reference Guide, March 1996.

[Alt96b] Alta Group of Cadence Systems, Inc. BONeS Designer, Application Library Reference Release 3.6, March 1996.

[Far95] FAROOQUI, Kazi; Logrippo, Luigi and de Meer, Jan. The ISSO Reference Model for Open Distributed Processing: An Introduction. Computer Networks and ISDN Systems 27 (1995) 1215-1229, 1995.

[Jus88] JUSTO, George Roger Ribeiro. Ambiente de Programação Distribuída com Configuração Dinâmica de Processos. Dissertação de Mestrado, UFPE, Setembro 1998.

[Kir95] KIRSTE, Thomas. An Infrastructure for Mobile Information Systems Based on a Fragmented Object Model. Computer Graphics Center Wilhelminenstr, January 1995.

[Mit94] MITCHELL, Kenneth J. and BARCLAY, Peter J. A Notation Independent Object Modelling Environment. Computer Studies Department, Napier University, 1994.

[Pal96] PALMA, Giovanny F. Lucero; CUNHA, Paulo R. Freire e MEIRA, Silvio Lemos. A LOTOS Specification of a CDMA Cellular System: Report of use. Departamento de Informática, Universidade Federal de Pernambuco, Agosto 1996.

[Pre95] PRESSMAN, Roger S. Engenharia de *Software*. Makron Books, 1995

[Pro95] Protosoft Inc. Paradigm Plus v3.0b, Reference Manual. Release 3.0 for UNIX and Windows Workstations, 1995.

[Que94]QUEIROZ, José Antônio Monteiro e CUNHA, Paulo Roberto Freire de. Sistemas Distribuídos: De Especificações LOTOS a Implementações. IX Escola de Computação, Julho 1994.

[Que94] QUEIROZ, José Antônio Monteiro e CUNHA, Paulo Roberto Freire . Sistemas Distribuídos: De Especificações LOTOS a Implementações. IX Escola de Computação, Julho de 1994.

[Ram95] RAMOS, Suzana de Queiroz; CUNHA, Paulo Roberto Freire e OLIVEIRA, Mauro. Uma Metodologia de Apoio à Análise e ao Desenvolvimento de Aplicações em Sistemas de Gerenciamento de Redes de Computadores. XIII Simpósio Brasileiro de Redes de Computadores, Belo Horizonte, Maio 1995.

[Ros96] ROSA, Nelson Souto. Implementando Projetos Formais LOTOS na Plataforma de Distribuição ANSAware. Dissertação de Mestrado, UFPE, 1996.

[Rum94] RUMBAUGH, James; BLAHA, Michael; PREMERLANI, William; EDDY, Frederick e LORENSEN, William. Modelagem e Projetos Baseados em Objetos. Campus, 1994.

[Soa90] SOARES, Luiz Fernando Gomes. Modelagem e Simulação Discreta de Sistemas. VII Escola de Computação, 1990.

[Som89] SOMERVILLE, I. Software Engineering, Addison-Wesley, 1989.

[Sou96] SOUZA, Adriano Augusto. Uma Extensão da Plataforma de Distribuição ANSAware para Suportar a Computação Móvel. Dissertação de Mestrado, Universidade Federal de Pernambuco, Dezembro 1996.