Modelagem de Sistemas Operacionais Utilizando LOTOS*

Cléver Ricardo Guareis de Farias
Wanderley Lopes de Souza
Célio Estevan Moron

Grupo de Sistemas Distribuídos e Redes (GSDR)
Departamento de Computação (DC)
Universidade Federal de São Carlos (UFSCar)
Caixa Postal 676 - CEP 13565-905 - São Carlos (SP) - Brasil
e-mail:{clever, desouza, celio}@dc.ufscar.br

Resumo

A Técnica de Descrição Formal (TDF) Language of Temporal Ordering Specification (LOTOS) tem sido utilizada com bastante sucesso no projeto de sistemas distribuídos e protocolos de comunicação. Entretanto, os conceitos arquitetônicos e as construções de LOTOS são gerais o suficiente para serem empregados em outros tipos de sistemas. O objetivo principal deste trabalho é demonstrar como LOTOS pode ser utilizada na modelagem de Sistemas Operacionais. Neste sentido, este trabalho apresenta os resultados da modelagem de um Sistema Operacional Multiprogramado em LOTOS e como esta especificação foi validada através de um conjunto apropriado de ferramentas.

Abstract

The Formal Description Technique (FDT) Language of Temporal Ordering Specification (LOTOS) has been successfully used in the design of distributed systems and communication protocols. However, the LOTOS architectural concepts and constructions are general enough to be employed in the development of other kinds of systems. The main goal of this paper is to show how LOTOS can be usefull in the modelling of Operating Systems. In this sense this paper presents the results of a Multiprogramming Operating System modelling using LOTOS and how this specification was validated by means of an apropriated set of tools.

Palavras-Chave: Sistemas Operacionais, TDF LOTOS, Engenharia de Software.

1 Introdução

Na Engenharia de Software o ciclo de vida dos sistemas pode ser dividido em várias fases [1]. A Figura 1 mostra o desenvolvimento de um sistema começando com a fase de Definição dos Requisitos do Usuário, onde são descritas as suas necessidades. Na fase de Definição dos Requisitos de Software é explicitado o que o sistema fará para preencher os requisitos dos

* Trabalho realizado com o auxílio da FAPESP e do CNPq.

usuário, enquanto que na fase de Definição do Projeto Arquitetônico o sistema é especificado em termos de componentes e recursos. Vários refinamentos dessa especificação são produzidos (fase de Definição do Projeto Detalhado) até a obtenção de um produto final, que inclui a documentação do sistema, seu código fonte e seu código executável. As últimas fases correspondem à Transferência do Produto e a sua consequente Operação e Manutenção (adaptações às mudanças de requisitos).

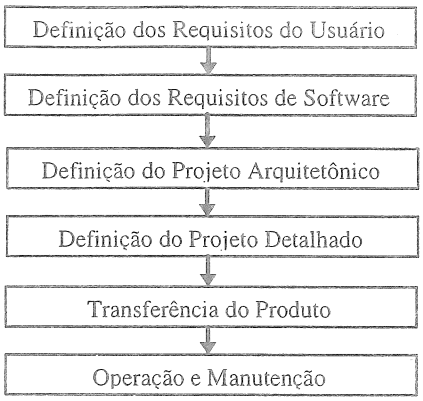

Figura 1 Fases do ciclo de vida de sistemas

O surgimento dos primeiros Sistemas Operacionais (SOs) [2,9] data dos anos cinquenta. Desde então os SOs vêm tornando-se cada vez mais complexos, a fim de atender aos mais variados requisitos de seus usuários. Esse aumento de complexidade frequentemente acarreta em erros nas fases iniciais do projeto, que se não forem detectados e corrigidos nessas fases serão transferidos para as fases posteriores, podendo vir a ser implementado um SO que não atenda aos requisitos propostos por seus usuários.

A maior parte desses erros ocorre principalmente devido ao emprego de métodos informais, que utilizam linguagens naturais, na fase de Definição dos Requisitos de Software. Além das ambiguidades intrínsecas a essas linguagens, tais métodos não permitem o uso de técnicas formais de validação, dificultam o trabalho em equipe e a padronização do produto. Consequentemente, é importante a adoção de uma metodologia, baseada em Técnicas de Descrição Formal (TDF), que facilite o desenvolvimento de SOs complexos.

Teoria das Filas [2], Cadeias de Markov [2], Redes de Petri [3], e a TDF Estelle [4] são algumas das técnicas que têm sido propostas para o projeto e o estudo do comportamento de SOs. A TDF Language of Temporal Ordering Specification (LOTOS) [5] tem sido utilizada com sucesso na especificação de vários tipos de sistemas, dentre os quais protocolos de comunicação [6], sistemas de lógica digital [7] e sistemas de telefonia [8]. Neste trabalho procura-se demonstrar como LOTOS pode ser utilizada na modelagem de sistemas operacionais, através da especificação de um Sistema Operacional Multiprogramado (SOM). A próxima seção descreve a arquitetura típica de um SOM, enquanto que a seção 3 apresenta um estudo de caso onde LOTOS

é empregada na modelagem de um SOM. Na seção 4 é discutida a validação da especificação produzida, enquanto que na última seção as conclusões são apresentadas.

2 Arquitetura típica de um SOM

A Figura 2 apresenta a arquitetura típica de um SO estruturado em camadas hierárquicas, onde cada camada, utilizando-se dos serviços oferecidos pela camada inferior, oferece um conjunto mais poderoso de serviços à camada superior. O sistema operacional UNIX [9] é um exemplo desse tipo de arquitetura, sendo que o Hardware e os Processos dos Usuários encontram-se respectivamente nas camadas inferior e superior, enquanto que o Kernel (gerenciamento de processos, gerenciamento de memória, sistema de arquivos, etc...) e Outras Funções do Sistema (editores, compiladores, interpretadores de comandos, etc...) encontram-se em camadas intermediárias.

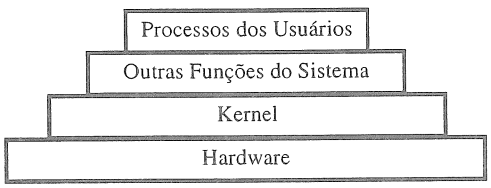

Figura 2 SO estruturado em camadas hierárquicas

Dentre as diversas funções desempenhadas por um SOM, destaca-se o gerenciamento de processos. Os processos são as entidades ativas de um SOM e quando criados têm suas principais informações (identificação do processo, prioridade, estado de execução, ponteiros para área de memória, etc...) guardadas em um Bloco de Controle de Processo (BCP). Este é um conceito fundamental na multiprogramação, pois é através dos BCPs que o SOM detém o controle dos processos que estão no sistema.

Um processo, representado por seu BCP, pode assumir diferentes estados durante sua existência, sendo que diversos tipos de eventos podem ocasionar mudanças de estado [2,9]. A Figura 3 apresenta um diagrama de transição de estados de um processo.

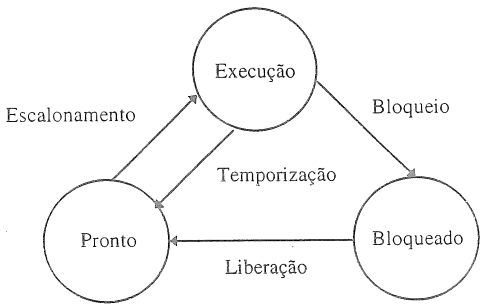

Figura 3 Diagrama de transição de estados

3 Especificação LOTOS de um SOM

LOTOS é uma das duas TDFs padronizadas em 1989 pela International Organization for Standardization (ISO) para a especificação formal de sistemas distribuídos e protocolos de comunicação, em particular aqueles relativos ao modelo Open System Interconnection (OSI). LOTOS é baseada em métodos algébricos tanto para a descrição de comportamentos quanto para a descrição de estruturas de dados e expressões de valores. Um sistema é modelado em LOTOS através de um ou mais processos contendo portas. Um processo é uma entidade abstrata capaz de interagir com o seu ambiente (outros processos LOTOS) através de ações observáveis e também capaz de realizar ações internas (não observáveis).

No nível mais abstrato, a arquitetura de um SOM pode ser modelada como a composição paralela de seus usuários, representados pelas várias instâncias do processo *Users*, com o sistema operacional propriamente dito, representado pelo processo *OperatingSystem*. A Figura 4 apresenta a arquitetura do primeiro nível da especificação LOTOS de um SOM.

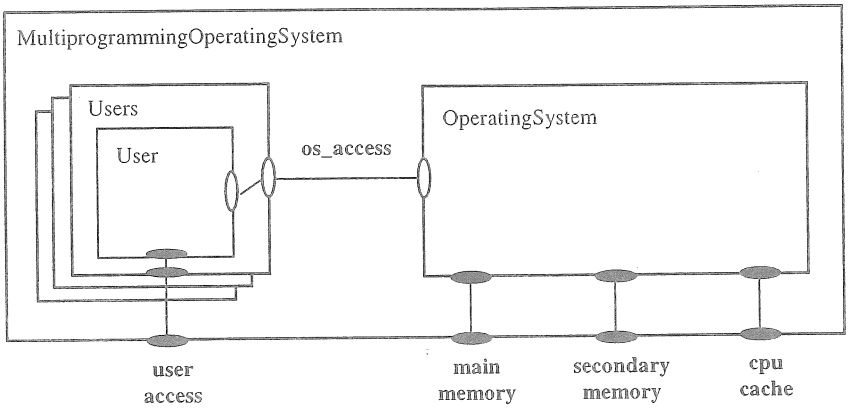

Figura 4 Arquitetura LOTOS do nível mais abstrato de um SOM

Users cria uma instância de *User*, que representa o comportamento de um usuário do sistema, reconstituindo-se para gerar novas instâncias dos usuários em paralelo, caracterizando o aspecto multiusuário do sistema. Cada usuário interage com sua instância de *User* através da porta *user_access*, podendo submeter processos, receber resultados da execução desses processos, receber mensagens de erro, etc.

3.1 *OperatingSystem*

Na especificação de *OperatingSystem*, identificamos dois conjuntos de funções distintas: o controle da submissão dos processos e o gerenciamento dos processos submetidos. Neste sentido, refinamos o processo *OperatingSystem* nos subprocessos comunicantes *Admitted* e *Processing*. *Admitted* armazena, na memória secundária, os processos admitidos no sistema, até que estes

possam ser carregados para a memória principal a fim de serem executados. *Processing* é o responsável pelo gerenciamento dos processos presentes no sistema segundo a filosofia da multiprogramação. A Figura 5 apresenta a arquitetura do processo *OperatingSystem*.

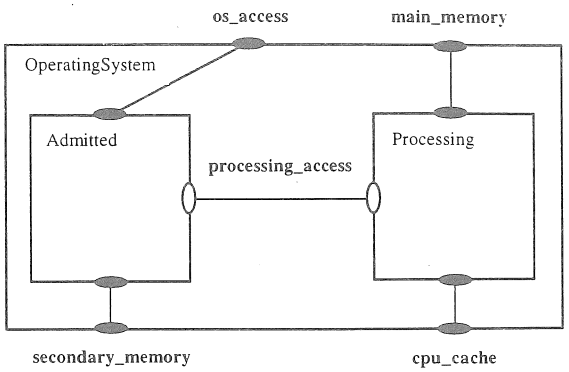

Figura 5 Arquitetura LOTOS de *OperatingSystem*

3.1.1 *Admitted*

Neste nível da especificação os seguintes conjuntos de restrições foram identificados: restrições referentes ao usuário, restrições referentes ao sistema, e restrições referentes ao gerenciamento da memória secundária. Neste sentido, refinamos *Admitted* nos subprocessos comunicantes *UserInterface, SystemInterface,* e *SecondaryMemoryManagement,* para atender a cada um destes conjuntos de restrições separadamente. A Figura 6 ilustra a arquitetura do processo *Admitted*.

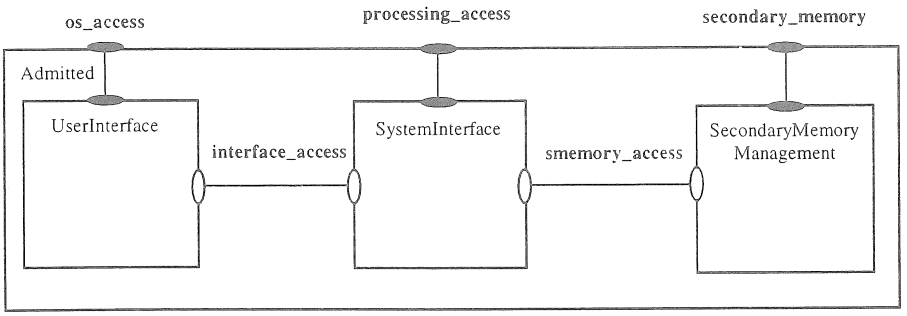

Figura 6 Arquitetura LOTOS de *Admitted*

O processo *UserInterface* descreve as restrições quanto ao gerenciamento da submissão dos processos que estão relacionadas diretamente com os processos dos usuários. Assim, refinamos este processo nos subprocessos comunicantes *ProcSubmissionHandler* e *ProcResultsHandler*. O processo *ProcSubmissionHandler* trata da recepção de um processo submetido ao sistema e da notificação, para o usuário, de aceitação ou não deste processo. O envio dos resultados da execução de um processo para os usuários é função do processo *ProcResultsHandler*.

O processo *SecondaryMemoryManagement* gerencia a memória secundária. Na modelagem LOTOS do SOM, um processo do usuário ao ser criado é inicialmente armazenado na memória secundária e posteriormente, em função da carga do sistema, é transferido para a memória principal. Dessa forma, dentre as funções desse processo, destacam-se o armazenamento na memória secundária dos processos recém criados pelos usuários e dos processos que sofreram permuta.

O processo *SystemInterface* funciona como uma interface entre os processos de *Admitted, UserInterface* e *SecondaryMemoryManagement*, e o processo *Processing*. *SystemInterface* foi refinado nos subprocessos independentes *ProcessCreationHandler* e *ProcessingResultsHandler*. O processo *ProcessingResultsHandler* recebe os resultados da execução dos processos e os envia para *UserInterface*. O processo *ProcessCreationHandler* por sua vez possui basicamente dois conjuntos de funções distintos: o recebimento de solicitações para a criação de novos processos (diretamente dos usuários ou de um processo em execução) e o recebimento/envio de processos que sofreram permuta.

3.1.2 *Processing*

Processing é responsável pela implementação da multiprogramação propriamente dita (veja Figura 3). Refinamos este processo em três subprocessos comunicantes (*Waiting, Running e Ready*), representando os possíveis estados de um processo no esquema geral da multiprogramação: bloqueado, em execução e pronto, respectivamente. A Figura 7 apresenta a arquitetura do processo *Processing*.

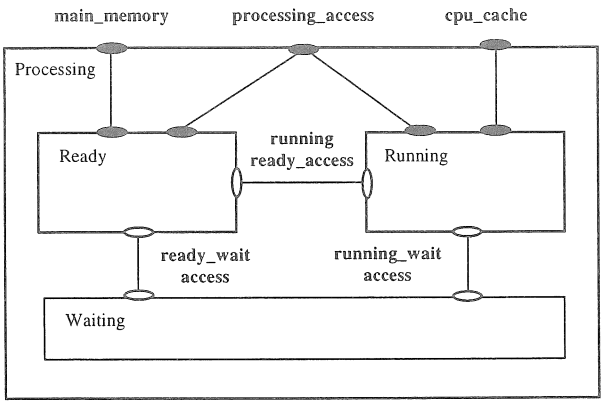

Figura 7 Arquitetura LOTOS de *Processing*

A fim de garantir a exclusão mútua no acesso a um recurso do SOM, quando um processo dos usuários deseja ter acesso a um recurso e este está sendo utilizado, este processo deve ser bloqueado. Tais processos são associados a eventos, que indicam a liberação do recurso aguardado, e colocados numa fila em *Waiting*. À medida em que os recursos são liberados, o primeiro processo na fila, que esteja associado a esse recurso, é transferido para *Ready*.

Running descreve o comportamento de um processo do usuário em execução dentro da CPU. Este processo possui dois estados: *idle*, indicando que a CPU está ociosa; e *busy*, indicando que há um processo sendo executado. Sempre que *Running* está ocioso (estado igual a *idle*), é requisitado um novo processo para execução. Outros eventos relacionados com a execução de um processo e descritos na especificação são: a emissão de resultados do processamento; a criação de um processo filho; a requisição de memória; a indicação do término da fatia de tempo; o término de execução; o bloqueio por falta de um recurso que está sendo utilizado e a sinalização do término de um evento.

3.1.2.1 Ready

Ready é o responsável pelo gerenciamento tanto da fila de processos no estado pronto quanto da memória principal. Para atender a esses requisitos separadamente, refinamos este processo nos subprocessos comunicantes *ReadyQueueHandler* e *MainMemoryManagement*. A Figura 8 apresenta a arquitetura do processo *Ready*.

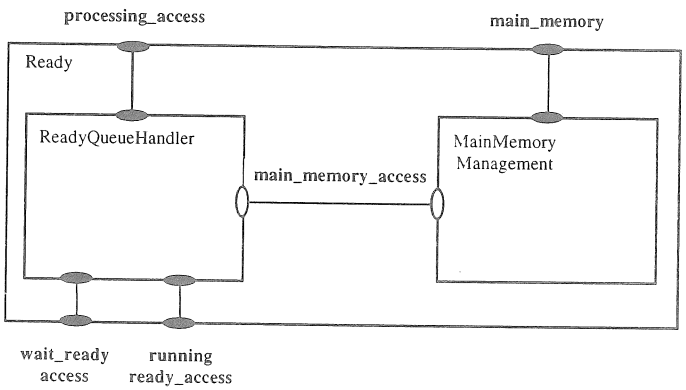

Figura 8 Arquitetura LOTOS de *Ready*

ReadyQueueHandler é o responsável pelo gerenciamento da fila de processos prontos. Dentre suas atribuições destacam-se: o envio/a recepção de um processo para/de *Running*, a recepção dos processos que estavam bloqueados e em disco, e a permuta de processos para o disco.

MainMemoryManagement é o responsável pelo gerenciamento da memória principal. Dois aspectos foram levados em consideração na modelagem deste processo: a necessidade do controle do grau de utilização da memória e a realização das operações de memória propriamente ditas. Baseando-nos nestas restrições, refinamos o processo *MainMemoryManagement* nos subprocessos independentes *MemoryOperation* e *MemoryStatus*.

MemoryOperation é o responsável pela realização das operações de memória tais como: o carregamento de um processo do disco para a memória principal; a liberação da memória quando um processo termina ou sofre permuta e a alocação de novas áreas de memória. *MemoryStatus* é o responsável pelo controle da utilização da memória. Este processo periodicamente recebe

informações sobre a área de memória disponível e baseando-se nessas informações decide se novos processos podem ser carregados para a memória principal ou se processos que estão carregados na memória devem sofrer permuta.

4 Validação da Especificação

Esta seção apresenta uma breve descrição da abordagem utilizada na validação da especificação LOTOS do SOM. Para a validação deste estudo de caso, os conjuntos de ferramentas MiniLite (versão 1996) [10] e Meije (versão 1995) [11] foram utilizados em uma abordagem combinando simulação e verificação.

A primeira etapa da validação tratou dos Tipos Abstratos de Dados (TADs) da especificação. Duas ferramentas foram utilizadas nesta etapa: o ADT Report [12] e o ADT Interface [12]. O ADT Report foi utilizado para prover consistência às definições dos TADs. Através do uso desta ferramenta, alguns axiomas críticos, que poderiam acarretar em comportamentos inesperados, foram identificados e substituídos por axiomas consistentes. Já o ADT Interface, parte integrante do Smile [12], o simulador simbólico interativo do MiniLite, foi intensamente utilizado na simulação dos TADs da especificação do SOM. Alguns erros foram encontrados e corrigidos através desta ferramenta.

Após a validação dos TADs da especificação, o próximo passo foi a validação do comportamento da mesma. Esta atividade de validação teve como propósito garantir que a especificação LOTOS foi produzida de acordo com a descrição informal de um SOM. Neste sentido, a primeira etapa foi a simulação de cada processo da especificação usando o Smile. Durante esta fase, procuramos garantir que cada processo foi especificado de forma correta, independentemente das especificações dos demais processos. Primeiramente simulamos os processos mais simples, e paulatinamente partimos para a simulação de processos mais complexos, até que a especificação como um todo fosse simulada. Durante esta fase, várias inconsistências e impasses foram identificados e corrigidos.

A segunda abordagem, utilizada na validação das expressões comportamentais da especificação, foi a execução da especificação em paralelo com um processo Teste. Este processo continha várias sequências de testes, modelando diferentes cenários de simulação, que deveriam ser preenchidas para garantir o sucesso do teste. Vários cenários foram testados, tais como: a submissão de processos, o término de processos, o envio de resultados do processamento, situações de sobrecarga de processamento e bloqueio/liberação de processos. Durante esta etapa de validação, apenas uns poucos erros foram encontrados e corrigidos.

Finalmente, a última abordagem, relacionada com a validação do comportamento da especificação, foi a geração e a análise da Máquina de Estados Finita Estendida (MEFE) da especificação. Utilizando-se o Smile, é possível gerar a MEFE completa de um dado processo. Esta máquina de estados é então transformada em um código de representação de autômatos, chamado FC2, que é a entrada para as ferramentas de verificação Meije. Duas ferramentas do

conjunto Meije foram utilizadas neste trabalho: Mauto [11], uma ferramenta para a verificação comportamental, e Autograph [11], um gerador de representações gráficas de autômatos.

Após a transformação da MEFE de um processo no código FC2, este código é reduzido por bissimulação fraca através do Mauto. Algumas propriedades deste autômato reduzido, por exemplo a existência de impasses, foram verificadas e, quando pequeno o suficiente, o autômato foi visualizado e analisado usando o Autograph. Esta metodologia foi aplicada para cada processo da especificação. Durante esta atividade nenhum erro ou impasse foi encontrado. Devido às limitações de memória, e ao problema de explosão de estados, foi impossível gerar o autômato para a especificação como um todo.

5 Conclusão

Este trabalho apresentou uma abordagem para a modelagem de sistemas operacionais baseada na utilização da TDF LOTOS nas primeiras fases do ciclo vida desses sistemas. Essa abordagem foi empregada na modelagem de um sistema operacional multiprogramado, sendo que uma especificação LOTOS foi produzida alternando-se os estilos de especificação orientado a recursos e orientado a restrições [13].

A especificação LOTOS do SOM pode ser encontrada em [14] e possui aproximadamente 830 linhas de código, dentre as quais 50% foram utilizadas na especificação dos Tipos Abstratos de Dados e 50% na especificação das expressões comportamentais. Comparando-se com a especificação Estelle apresentada em [4], percebemos que a utilização de LOTOS resultou em um código sensivelmente mais conciso para a especificação de um mesmo sistema, excluindo-se as devidas diferenças.

Contudo, a principal vantagem da utilização de LOTOS no modelagem de um SOM, foi a possibilidade de se descrever um mesmo sistema em diferentes níveis de abstração, sem que fosse necessário entrar em detalhes relativos às implementações desse sistema. Dessa forma, um determinado componente pode ser estudado com mais ou menos detalhes em função da ênfase utilizada na modelagem. Isso possibilita, por exemplo, a especificação de vários tipos de comportamentos para um mesmo componente, a análise desses comportamentos, e a utilização do comportamento mais adequado a cada tipo de situação (e.g., diferentes políticas de escalonamento).

Várias ferramentas que dão suporte a maior parte do ciclo de vida dos sistemas de comunicação já foram desenvolvidas para LOTOS. Em particular, o ambiente MiniLite e o conjunto Mauto/Autograph foram utilizados no desenvolvimento e validação da especificação do SOM. Uma abordagem que combinou técnicas de simulação e verificação foi empregada na validação da especificação produzida. Esta abordagem possui algumas limitações, pois além de utilizar cenários pré-definidos e limitados nos testes da especificação, não foi possível gerar o autômato para a especificação como um todo, impossibilitando assim, afirmar que a especificação não possua impasses. Entretanto esta abordagem confere uma grande confiabilidade à especificação

produzida à medida em que se procura identificar e corrigir os erros presentes na especificação através de um conjunto de técnicas aplicáveis a sistemas reais e complexos. O que nem sempre é possível com outras técnicas de validação.

Referências

[1] J. Quemada, A. A. Saloña, S. P.Gómez. *Design with LOTOS*, Technical Report DIT-UPM, Junho/1993.

[2] H.M.Deitel. *Operating Systems, Second Edition*. Addison-Wesley, Massachusetts, 1990.

[3] G.W. Brams. *Réseaux de Petri: Théorie and Pratique*, T.1 y 2. Ed. Masson, Paris, 1986.

[4] Gonzalez, O, et Al. *Design of operating systems using the FDT ESTELLEe*. Dieter Hogrefe e Stefan Leue (Ed), VII International Conference on Formal Description Techniques, pp 51-66, 1994.

[5] ISO IS 8807. *Information processing systems - Open Systems Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour*, 1989.

[6] I. Ajubi. *Formal Description of the OSI Session Layer: Session Protocol*. No livro The Formal Description Technique LOTOS. P.H.J van Eijk, C.A. Vissers, M. Diaz (eds), North-Holland, pp.153-210, 1989.

[7] K.J. Turner, Richard O. Sinnott. *DILL: Specifying Digital Logic in LOTOS*. In Richard L. Tenney, Paul D. Amer, and M. Umit Uyar (eds), Proc. Formal Description Techniques VI, North-Holland, Amsterdam, the Netherlands, 1994, pp 71-86.

[8] R. Boumezbeur e L. Logripo. *Specifying Telephone System in LOTOS*, IEEE Communications Magazine, agosto de 1993.

[9] A. S. Tanembaum. *Modern Operating Systems*. Prentice-Hall, 1992.

[10] T. Bolognesi, J. v.d. Lagemaat, C. A. Vissers (editors), *LOTOSphere: Software development with LOTOS*, Kluwer Academic Publishers, 1995.

[11] E. Madelaine, *Verification Tools from the CONCUR Project*. Bulletin of the EATCS, 47:110-127, 1992.

[12] M. Caneve, E. Salvatori (CPR)(eds), *Lite User Manual: Final Deliverable*, Lo/WP2/N0034/V08. ESPRIT Ref: 2304, 1992.

[13] C.A. Vissers, G. Scollo, M. van Sinderen e E. Brinksma. *On the use of specification styles in the design of distributed systems*. In P.H.J.van Eijk, C.A.Vissers e M.Diaz (Eds), The Formal Description Techinique LOTOS. North-Holland, 1989.

[14] C. R. G de Farias, W. L. de Souza, e C. E. Moron. *Especificação Formal e Validação de um Sistema Operacional Multiprogramado*. Relatório Técnico No. 004/97, DC/CCT/UFSCar, São Carlos (SP), 1997.