Universidade Federal de Santa Catarina

Curso de Pós-Graduação em Ciências da Computação
Sistemas de Computação - Especificação de Sistemas

Projeto de Sistemas Utilizando Construções Predefinidas Aplicadas à Gerência de Redes: a Biblioteca bibLOTOS

Dissertação submetida
à Universidade Federal de Santa Catarina
para a obtenção do grau de
Mestre em Ciências da Computação.

Cristiano Maciel

Florianópolis, 26 de fevereiro de 1996.

Projeto de Sistemas Utilizando Construções Predefinidas Aplicadas à Gerência de Redes: a Biblioteca bibLOTOS

Cristiano Maciel

Esta dissertação foi julgada adequada para a obtenção do título de

MESTRE EM CIÊNCIAS DA COMPUTAÇÃO

na área de concentração Sistemas de Computação, sub-área Especificação de Sistemas (Especificação Formal) e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciências da Computação da UFSC.

Bernardo Gonçalves Riso - Orientador

Murilo Silva de Camargo - Coordenador do CPGCC/UFSC

Banca Examinadora:

Bernardo Gonçalves Riso

Carlos Becker Westphall

Aloysio de Castro Pinto Pedroza

João Bosco Mangueira Sobral

Ronaldo dos Santos Mello (suplente)

Dedico este trabalho, com amor,
à minha mãe,
Teresinha Carmem Maciel.

AGRADECIMENTOS

Inicialmente, gostaria de agradecer aos professores integrantes da banca examinadora pela apreciação do presente trabalho, assim como pelas críticas e sugestões, que contribuiram para o aperfeiçoamento desta dissertação.

Também gostaria de agradecer aos professores que, ao longo do curso, contribuíram com informações de toda ordem para a realização deste trabalho.

Agradeço igualmente à equipe do Laboratório de Redes e Gerência, não só pela estrutura física e recursos disponibilizados, mas também a todos os colegas, em especial à Mirela Sechi Moretti Annoni Notare e à Analúcia Schiaffino Morales De Franceschi, pela amizade e ajuda prestada.

Agradeço aos colegas da Pós-Graduação e aos amigos que, durante esta caminhada, fizeram-se presentes, participando dos acontecimentos, enfim, vivendo o mestrado. Em especial agradeço aos colegas Henrique e Patrícia e aos companheiros de Jurerê pela companhia constante nessa caminhada.

Reconheço minha gratidão ao meu orientador, professor Bernardo Gonçalves Riso, através do qual e com quem todo este trabalho foi possível.

Ao CPGCC/UFSC pela oportunidade de realização do mestrado, as secretárias da pós-graduação Verinha e Valdete pelo apoio e amizade, e à CAPES pela bolsa oferecida durante o período de elaboração da dissertação.

A todos, muito obrigado.

ÍNDICE

LISTA DE FIGURAS

LISTA DE SIGLAS

ADT	*Abstract Data Type*
ADT	*Access Developer's Toolkit*
ASN.1	*Abstract Sintaxe Notation - Number One*
CADP	*Cæsar Aldébaran Distribution Package*
CAPES	Coordenação de Aperfeiçoamento de Nível Superior
CCITT	*Consultative Commitee of International Telegraph and Telephone*
CCS	*Calculus of Communicating Systems*
CMIP	*Common Management Information Protocol*
CMIS	*Common Management Information Service*
CMISE	*Common Management Information Service Element*
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CPGCC	Curso de Pós-Graduação em Ciências da Computação
CTC	Centro Tecnológico
ELTS	*Extended Labeled Transition System*
ESTELLE	*Extended Finite State Transition Language*
EUCALYPTUS	*European-Canadian LOTOS Protocol Tool Set*
FDT	*Formal Description Technique*

GDMO	*Guidelines for the Definition of Managed Objects*
G-LOTOS	*Graphical Language of Temporal Ordering Specification*
IBM	*International Business Machines*
INE	Departamento de Informática e de Estatística
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
ISODE	*ISO Development Environment*
ITU	*International Telecommunication Union*
LITE	*LOTOSphere Integrate Tools Environment*
LOTOS	*Language Of Temporal Ordering Specification*
LOWE	*LOTOS Windows Environment*
LRG	Laboratório de Redes e Gerência
LTS	*Labeled Transitions System*
MIB	*Management Information Base*
MOs	*Managed Objects*
OLE	*Object Linking and Embeding*
OSI	*Open System Interconnection*
OSIMIS	*OSI Management Information Service*
PC	*Personal Computer*

PDICs	*Pre-Defined Implementation Constructs*
PLAGERE	Plataformas para Gerência de Redes
PROTEM-CC-II	Programa Temático Multi-Institucional - Ciência da Computação-Fase II
RMON	*Remote Monitoring Network*
SDL	*Specification and Description Language*
SGBD	Sistema Gerenciador de Banco de Dados
SNMP	*Simple Network Management Protocol*
TCP	*Transmission Control Protocol*
TDF	Técnica de Descrição Formal
TOPO	*LOTOS compoiler*
TranSP	*Transformator From Service To Protocol*
UFSC	Universidade Federal de Santa Catarina

Resumo

Neste trabalho apresenta-se uma abordagem para o projeto de sistemas de gerência de redes que emprega uma biblioteca (biblioteca bibLOTOS), de construções predefinidas, realizadas com o uso da Técnica de Descrição Formal LOTOS. A biblioteca inclui construções predefinidas em LOTOS Básico (onde apenas os aspectos de comportamento são definidos), construções predefinidas em ACT ONE (onde são considerados os aspectos de dados) e construções predefinidas em LOTOS Completo (onde, além dos aspectos de comportamento, também são considerados os aspectos de dados). O método proposto sugere que, durante a concepção inicial do sistema, bem como durante os refinamentos sucessivos deste, o projetista resgate as construções predefinidas da bibLOTOS, que são de interesse, e as integre à descrição do sistema em desenvolvimento. Para dar suporte ao método proposto um ambiente de desenvolvimento foi concebido e implementado. O ambiente LOWE (*LOTOS Windows Environment*) está implementado em microcomputadores IBM PC e é composto pelas ferramentas: edLOTOS, bibLOTOS e TranSP. A biblioteca bibLOTOS é implementada utilizando o sistema *Access* para microcomputadores. O emprego da bibLOTOS assim como do método a ela associada são ilustrados com o projeto de um sistema de gerência proativa de redes. A gerência proativa visa prever os problemas que podem ocorrer na rede e evitar que eles degradem os serviços oferecidos aos usuários. O agente proativo tem seu comportamento especificado em LOTOS, de acordo com o método proposto, partindo de um nível mais alto de abstração e sendo sucessivamente refinado. Ferramentas LOTOS são utilizadas para a validação das construções predefinidas presentes na bibLOTOS bem como das especificações.

Abstract

This work presents an approach for the design of management systems through the usage of a library (bibLOTOS) of pre-defined constructions which are specified in LOTOS Formal Description Technique. The library encompasses pre-defined constructions in Basic LOTOS (only behavioural aspects are defined), in ACT ONE (only data aspects are considered) and also in Full LOTOS (where both behavioural and data aspects are considered). The methodology here proposed suggests that in the initial design of the system as well as in its subsequent refinements, the developer search the bibLOTOS pre-defined constructions of interest and incorporate them to the specification of the system under development. A development environment was designed and implemented in order to provide support to the proposed methodology. The *LOTOS Windows Environment - LOWE* - is implemented for IBM PC and is composed by the edLOTOS, bibLOTOS and TranSP tools. The bibLOTOS library is implemented using the Access system for PCs. The usage of bibLOTOS as well as the respective methodology are illustrated through the design of the system for network proactive management. Proactive management aims at identifying the existing troubles in advance to any performance degradation on the services provided to the users. The behaviour of the proactive agent was specified in LOTOS adopting the proposed methodology, initially in a high level of abstraction and followed by a stepwise refinement. LOTOS tools were employed to validate the pre-defined constructions present in the bibLOTOS and the resultant specifications.

Palavras-chave

Sistemas distribuídos

Serviços de comunicação

Protocolos de comunicação

Gerência de redes

Gerência proativa

Especificação formal

Construções predefinidas

Reutilização de especificações

Ferramentas de projeto

bibLOTOS

LOTOS

Keywords

Distributed systems

Communication services

Communication protocols

Network management

Proactive Management

Formal specification

Pre-defined constructs

Specification reuse

Design tools

bibLOTOS

LOTOS

Apresentação

Vários setores industriais servem-se de produtos pré-fabricados (de uso geral) para a manufatura de novos produtos (de uso específico). Esse é o caso, por exemplo, da indústria de construção civil onde tijolos, tubos, armações, blocos, vigas, lages e muitos outros produtos pré-fabricados são empregados na construção de edifícios. Exemplos na indústria de alimentos (que emprega alimentos pré-elaborados) e em outras indústrias (como é o caso da indústria eletro-eletrônica) não faltam. O sucesso obtido por todas essas indústrias no que se refere ao aceleramento do processo de produção e redução dos custos é inegável.

Pois bem, as empresas de produção de *software* e, em particular, as empresas de produção de *software* de comunicação (aí incluídos os protocolos de comunicação para as redes de computadores e para as redes de telecomunicações) podem adotar uma abordagem semelhante àquela já utilizada pela indústria. Essa abordagem consiste em empregar, nos seus projetos, um conjunto de construções predefinidas de uso geral e reutilizáveis.

Essa idéia não é nova. Entretanto, hoje estabelecem-se condições quase ideais para concretizá-la. Essas condições consistem na disponibilidade de técnicas de descrição formal (TDF's) padronizadas (e, portanto, estáveis) e, associadas a essas TDF's, métodos de projeto que se apóiam na utilização de poderosas ferramentas conceituais e automáticas (dando ao processo de desenvolvimento um cunho sistemático).

Tais condições, uma vez reunidas, constituem um arsenal de grande efeito para o *design* e a implementação de sistemas de comunicação (por exemplo, na área de gerência de redes) com

rapidez, segurança e baixo custo de desenvolvimento. Desse arsenal uma biblioteca de construções predefinidas é um dos elementos principais.

Para dar ao processo de desenvolvimento de *software* de comunicação e de gerência de redes uma feição industrial é preciso, portanto, montar tal biblioteca de construções predefinidas e estabelecer um método de projeto que oriente o uso sistemático dessa biblioteca em projetos específicos. Esse é o objetivo do presente trabalho.

Este trabalho dá continuidade a uma série de pesquisas em torno da utilização de técnicas de descrição formal para o projeto de sistemas distribuídos em geral e, em especial, para o projeto de sistemas de gerência de redes. Em trabalhos anteriores dessa série foram investigados métodos baseados em refinamentos sucessivos (com o uso de ferramentas disponíveis internacionalmente) e métodos baseados em transformações automáticas (nesse caso uma nova ferramenta, a ferramenta TranSP, foi desenvolvida).

Para tal, na Universidade Federal de Santa Catarina (UFSC) um grupo de pesquisadores ligados ao Laboratório de Redes e Gerência (LRG) vem se empenhando no estudo e no desenvolvimento de sistemas para a gerência de redes. Esse grupo, que inclui professores, estudantes de graduação e pós-graduação, além de bolsistas do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), está envolvido em vários projetos. Para dar uma idéia precisa do ambiente de pesquisa no qual foi desenvolvido o presente trabalho, e dos temas que estão sendo objeto de investigação, apresentam-se, a seguir, as características do projeto mais importante ao qual está ligado o LRG: o projeto PLAGERE.

O projeto PLAGERE - Plataformas para Gerência de Redes (Processo nº 68.072/94-2), que envolve equipes de pesquisa do CPqD-TELEBRÁS (Campinas - SP), CEFET-PR (Curitiba-PR) e Universidade Federal da Paraíba (Campina Grande-PB) sob a coordenação geral do professor Carlos Becker Westphall da UFSC (Florianópolis-SC), é um projeto do Programa Temático Multi-institucional em Ciência da Computação - Fase II (PROTEM-CC-II/CNPq). Esse projeto tem como objetivo a realização de estudos e propostas ligadas a plataformas e aplicações para plataformas de gerência de redes de computadores e telecomunicações.

Para auxiliar na realização das tarefas previstas nesses projetos diversas ferramentas automáticas estão instaladas no LRG. No contexto da gerência de redes destacam-se o *SunNet Manager*, o OSIMIS e o ISODE, bem como a ferramenta de simulação ARENA. O uso da TDF LOTOS é apoiado pelas ferramentas EUCALYPTUS, CADP e TOPO.

Com este trabalho o mestrando Cristiano Maciel apresenta uma proposta que complementa realizações anteriores, e harmoniza-se com elas, estabelecendo novos recursos para a abordagem de projetos complexos. Tais recursos, reunidos àqueles já desenvolvidos em trabalhos anteriores, (e aliados a uma cultura de investigação e desenvolvimento apoiada no uso de ferramentas automáticas) compõem a infraestrutura necessária para a formação de um escritório de engenharia de protocolos, voltado para a produção industrial de sistemas na área de gerência de redes, em parceria com outras instituições de pesquisa e organizações da iniciativa privada.

Bernardo Gonçalves Riso

Orientador

1. Introdução

"The software environment, including automated facilities, is the implementation of development policies and practices" [Evan 89].

Sistemas distribuídos de interesse prático são, normalmente, sistemas complexos e de grande porte. O projeto de tais sistemas é, por esses motivos, difícil de realizar. Para que o projeto e a implementação de sistemas distribuídos realizem-se de modo rápido e seguro pode ser conveniente a utilização de técnicas de engenharia que empreguem métodos formais.

O uso de técnicas formais no projeto e no desenvolvimento de sistemas complexos contribui para dar rigor, precisão e concisão ao projeto, além de permitir a análise e a implementação de sistemas com o uso de ferramentas automáticas. Uma técnica de descrição formal (TDF) que vem sendo utilizada no projeto de sistemas distribuídos é a TDF LOTOS [ISO 8807], um padrão internacional da ISO (*International Organization for Standardization*).

Para a linguagem LOTOS foram sugeridas diferentes abordagens de utilização. Uma dessas abordagens considera a possibilidade do uso de construções predefinidas como um recurso de projeto [PiSi 92].

Ao permitir a reutilização de especificações, o emprego de construções predefinidas pode reduzir os esforços para a validação de sistemas, proporcionando um desenvolvimento mais rápido [PiSi 92] [SiPi 92]. O propósito principal do uso de construções predefinidas é obter uma especificação de protocolo, ou de outro sistema, do modo mais rápido possível, encurtando a trajetória de desenvolvimento destes [ScPi 92] [Nota 95]. Outros propósitos são a padronização das especificações e a facilidade de manutenção dos sistemas desenvolvidos.

Diversos sistemas distribuídos e protocolos de comunicação podem ser desenvolvidos e implementados utilizando um dado conjunto de construções predefinidas. Tais construções podem estar disponíveis em um banco de dados para fácil acesso e reutilização.

Uma biblioteca (bibLOTOS) foi implementada em um ambiente integrado de ferramentas baseadas na TDF LOTOS (*LOWE: LOTOS Windows Environment*) a fim de armazenar as construções predefinidas que auxiliam o desenvolvimento de especificações [MaNo 96a] [MaNo 96b]. Na bibLOTOS as construções predefinidas estão acompanhadas de um conjunto de informações adicionais (como por exemplo: descrição informal da construção, sua representação gráfica, código C correspondente, entre outras informações) que auxiliam na escolha da construção predefinida que melhor atenda as necessidades do projetista. Durante a construção da especificação de um sistema a inclusão das construções predefinidas é feita através de um editor (edLOTOS) disponível no ambiente LOWE [MaNo 96c].

O método proposto sugere que as construções predefinidas armazenadas na biblioteca bibLOTOS sejam reutilizadas à medida que o projeto do sistema evolui. Durante o seu trabalho o projetista recolhe as construções predefinidas que são de seu interesse e as integra à descrição do sistema em desenvolvimento.

A busca de eficiência no processo de desenvolvimento, isto é, a preocupação com a definição de um método de efetivo uso prático, parece indicar que a aplicação de um dado conjunto de construções predefinidas deve-se dar em um domínio de aplicação restrito. Tendo isso em vista procurou-se definir e utilizar um conjunto de construções predefinidas no domínio de aplicação

ligado aos sistemas de gerência de redes de computadores e telecomunicações, e em particular, a gerência proativa desses tipos de redes.

Na continuidade deste capítulo introdutório apresentam-se, a seguir, as principais contribuições realizadas, as publicações e participações em feiras de tecnologia e em congressos e, finalmente, a organização geral do trabalho.

1.1 Contribuições desse trabalho

Este item tem por finalidade destacar as contribuições do presente trabalho no que se refere a aspectos de projeto na área de sistemas de gerência de redes com o uso de Técnicas de Descrição Formal.

Para identificar obras de autores que desenvolveram pesquisas nas áreas de gerência de redes, assim como propostas de métodos de projeto com a reutilização de especificações, inicialmente fez-se uma revisão de natureza bibliográfica. Esta revisão foi fundamental para o desenvolvimento do presente trabalho, pois serviu para obter conhecimentos preliminares acerca das áreas pesquisadas, estabelecer um histórico das investigações sobre a reutilização de especificações e definir algumas tendências nessa área.

Após elaborar uma crítica dos trabalhos existentes, este trabalho procura dar um passo além, pretendendo contribuir para a concepção de um método de projeto que reutilize especificações LOTOS. Dentro deste contexto, e neste trabalho, as especificações reutilizáveis são chamadas de construções predefinidas.

Pode-se dizer que a principal contribuição desse trabalho é a bibLOTOS - uma biblioteca de construções predefinidas, que aliada a um conjunto de informações úteis para a condução de projetos, visa auxiliar o projetista na tarefa de especificar sistemas complexos de gerência de redes. A biblioteca bibLOTOS, implementada em ACCESS, é uma ferramenta que pode ser instalada em microcomputadores de modo a dar comodidade ao projetista e contribuir para difundir o uso de Técnicas de Descrição Formal.

Visto que, para a reutilização eficiente de construções predefinidas, convém fixar um domínio de aplicação, estabeleceu-se a área de gerência de redes, tendo como exemplo de aplicação neste domínio a gerência proativa de redes de computadores. Neste domínio foram definidas cento e quarenta construções predefinidas em LOTOS Básico, ACT ONE e LOTOS Completo.

A participação do autor no projeto PLAGERE - PROTEM-CC-II/CNPQ foi de muita valia durante o desenvolvimento do trabalho. Através do convívio no LRG e na companhia do grupo de pesquisa, o autor esteve em contato constante com trabalhos da área de gerência de redes, informando-se continuamente e discutindo acerca destes trabalhos.

A participação como ouvinte em palestras, em defesas de trabalhos individuais e dissertações de mestrado e doutorado, bem como a apresentação e publicação de trabalhos junto a instituições nacionais e internacionais trouxe novos conhecimentos e serviu como instrumento de formação cultural e capacitação profissional.

1.2. Participações, Palestras, Feiras e Publicações

A participação em congressos científicos e feiras tecnológicas, assim como a realização de palestras e publicações contribuem para dar ao pesquisador a real dimensão dos trabalhos por ele desenvolvidos. Tendo isso em vista buscou-se essa participação e essas realizações durante a elaboração das investigações que culminaram com este trabalho.

Abaixo são relatadas as atividades referentes a participações em eventos, palestras ministradas e publicações realizadas durante o período de preparação desse trabalho.

1.2.1 Participações em defesas, painéis e congressos

Logo ao ingressar no curso de pós-graduação, no ano de 1995, o autor assistiu à defesa de dissertação intitulada "Uma metodologia para a especificação formal de serviços e protocolos de comunicação", de autoria de Mirela S.M.A. Notare, sob a orientação do professor Bernardo Gonçalves Riso. Tal defesa constituiu o primeiro contato do autor com um trabalho sobre especificação formal a nível de mestrado e despertou o interesse pelo presente tema.

Neste mesmo ano de 1995 o autor assistiu à defesa de dissertação da mestranda Cláudia Jacy Barenco, intitulada "Multimídia em redes: um estudo da transmissão de sinais de vídeo e audio em um backbone FDDI".

Durante o 2° Congresso de Tecnologia, Telecomunicações e Informática, o autor desta dissertação participou, como ouvinte, do Painel "Telecomunicações e Informática", ministrado pelos Srs. Antônio Sales Campos Filho (CPqD/ Campinas) e Angelo Fares Menhem (Funsoft/MG), promovido pela ADETEC de Londrina (PR), no dia 06/08/96. Os principais

assuntos do painel foram a Gerência de Redes de Telecomunicações (TMN) e a situação da indústria de *software* e das tecnologias de informação, tendo em vista as mudanças de paradigmas nas telecomunicações.

Com o objetivo de buscar informações e rever conhecimentos referentes à área de Redes de Computadores, além de fazer contatos com personalidades da comunidade científica que pudessem contribuir para o desenvolvimento desta dissertação, o autor participou, como ouvinte, do XIV Simpósio Brasileiro de Redes de Computadores - SBRC, realizado em Fortaleza (CE), no período de 20 a 23/05/1996. Aproveitando a oportunidade assistiu ao Minicurso intitulado "Qualidade de Serviço e Protocolos de Alta Velocidade", no dia 23/05/1996, que teve duração de três horas.

No decorrer do curso de pós-graduação o autor assistiu a defesas de Trabalhos Individuais, realizados pelos colegas, nas diferentes linhas de pesquisa do CPGCC/UFSC. Pode-se citar: 1) "Simulador para especificações formais de sistemas dependentes de tempo", de mestrando Roberto Milton Scheffel; 2) "Utilizando LOTOS na concepção formal de um *gateway* CMIP-SNMP para gerência de redes: especificação e verificação", do mestrando Bráulio Adriano de Mello; 3) "Gerência de Contabilização para provedores de serviço Internet", do mestrando Luiz Fernando Tavares Meirelles; e 4) "Sistema de autoria para *adventures* educacionais", da mestranda Patrícia Cristiane de Souza.

Também teve a oportunidade de assistir a apresentação de Trabalhos de Conclusão de Curso da graduação em Ciências da Computação da UFSC, a saber: 1) "Implementação de um elemento de rede (ponto de terminação de conexão) para a gerência de redes ATM usando o OSIMIS - TMN",

de autoria das alunas Clarissa Carneiro Mussi e Vivianne Noemi Fleith Mazorca; e 2) "Implementação de um elemento de rede (*trail termination point*) para a gerência de redes ATM usando OSIMIS - TMN", de autoria dos alunos Juliano Fontoura Pereira e Nelson A. Sinibaldi Filho. Estes trabalhos foram desenvolvidos no Laboratório de Redes e Gerência (LRG).

1.2.2 Palestras realizadas pelo autor

Em 29/03/96, na reunião mensal da SUCESU-SC, em Balneário Camboriú (SC), parte do grupo de pesquisa do LRG apresentou uma palestra com o título "*Projeto PLAGERE: Plataformas para Gerência de Redes*". Nesta palestra, inicialmente, o professor Fernando Augusto Cruz discorreu sobre algumas das principais plataformas de gerência de redes disponíveis atualmente no mercado. Logo após, o autor da presente dissertação apresentou aspectos da TDF LOTOS, bem como das ferramentas e abordagens de projeto, enfocando principalmente os aspectos relacionados ao uso de construções predefinidas. A palestra foi encerrada pelo professor Bernardo Gonçalves Riso, o qual apresentou uma visão geral dos trabalhos desenvolvidos pelo LRG, destacando a relevância destes trabalhos no processo de desenvolvimento tecnológico na área de projetos em gerência de redes.

O artigo "Usando LOTOS para descrever um sistema de consulta", de autoria de Josué da Luz Dias, Mirela S.M.A. Notare e Bernardo G. Riso foi apresentado por Cristiano Maciel no 2° CONTTEIN - Congresso de Tecnologia, Telecomunicações e Informática, realizado de 05 a 09/08/96, em Londrina (PR). Nesta palestra sugeriu-se a utilização da TDF LOTOS para representar o comportamento de um sistema de consulta, tendo como exemplo um Sistema de Consulta para Vídeo-Locadora. Para descrever o sistema foram adotadas duas abordagens.

Primeiramente o comportamento do sistema foi representado através de Diagramas de Fluxo de Dados, Português Estruturado e Árvores - recursos para análise do sistema. Posteriormente a TDF LOTOS foi utilizada para dar um tratamento formal ao projeto do sistema. Vantagens e desvantagens de cada abordagem foram sendo examinadas no decorrer da apresentação.

Em 15/10/1996 o autor teve a oportunidade de palestrar sobre "Análise de Sistemas e Especificação Formal", na Escola Cenecista Dr. João Dahne, em Santa Rosa (RS), tendo como público-alvo os alunos do curso Técnico em Processamento de Dados. A união dos conhecimentos adquiridos durante a graduação e pós-graduação viabilizaram uma palestra que serviu como estímulo ao público presente. A palestra teve duração de duas horas.

1.2.3 Participações em Feiras

De 05 a 09/08/96 realizou-se em Londrina (PR) a 5° INFOTECH - Feira de Informática e Tecnologia de Londrina, no Catuaí Shopping Center. O evento proporcionou ao autor desta dissertação um proveitoso contato com empresas e universidades expositoras de serviços e produtos. Cabe ressaltar os contatos firmados com a Associação de Desenvolvimento Tecnológico de Londrina, com o Núcleo Softex 2000 do Norte do Paraná e com a Universidade Estadual de Londrina.

No período de 2 a 6/12/1996 o mestrando Cristiano Maciel representou o Curso de Pós-Graduação em Ciências da Computação da UFSC como expositor junto ao *stand* das universidades, na área de exposição do III INFTEL - Congresso PETROBRÁS de Informática e Telecomunicações. Tal exposição constituiu em fornecer aos interessados informações sobre: 1)

o curso de pós-graduação, utilizando-se de cartazes, folders e tendo à disposição um micro-computador conectado à *Internet*, acessando assim a *Home-Page* do CPGCC/UFSC; 2) informações sobre o LRG e demonstração das ferramentas utilizadas nesse laboratório.

1.2.4 Publicações realizadas

O artigo intitulado "Usando a TDF LOTOS no projeto de protocolos de comunicação", de autoria de Luis F. Fausto, Cristiano Maciel e Bernardo G. Riso foi publicado nos anais do 2° CONTTEIN - Congresso de Tecnologia, Telecomunicações e Informática. Tal evento ocorreu em Londrina (PR), no período de 05 a 09/08/96. Neste trabalho são apresentados exemplos do uso da TDF LOTOS para o desenvolvimento de especificações de protocolos de comunicação, contribuindo para a obtenção de projetos bem elaborados. Esse trabalho tinha também como objetivo divulgar o uso de LOTOS no ambiente acadêmico e industrial. Apresentado por Cristiano Maciel.

O artigo "Biblioteca de Construções Predefinidas em LOTOS", de autoria de Cristiano Maciel, Mirela S.M.A. Notare e Bernardo G. Riso foi publicado nos anais da III Jornada de Pesquisa da UNIJUÍ no campus de Santa Rosa (RS), no período de 7 a 11/10/1996, na página 236. Neste trabalho apresenta-se uma abordagem para o projeto formal de sistemas de gerência de redes que emprega uma biblioteca de contruções predefinidas realizadas com o uso da TDF LOTOS. A biblioteca inclui construções em LOTOS Básico e construções em LOTOS Completo. A biblioteca é implementada utilizando o sistema ACCESS 2.0© para microcomputadores. Tal biblioteca busca dar suporte ao estabelecimento de um método de projeto e desenvolvimento de sistemas distribuídos que emprega a TDF LOTOS. Esse método é de uso geral mas, naquele

trabalho, ela é ilustrada através do projeto de um sistema de gerência proativa para redes de comunicação. Em tal trabalho considera-se que uma contribuição efetiva para o sucesso e a difusão de LOTOS consiste na disponibilização de ferramentas para PC, de fácil utilização, e integradas ao ambiente de desenvolvimento de especificações LOTOS. Apresentado por Cristiano Maciel.

O artigo "Projeto Formal de uma Plataforma para Gerência de Redes", de autoria de Elenirse Furlanetto , tendo como co-autores Cristiano Maciel, Mirella S.M.A. Notare e Bernardo G. Riso, foi publicado nos anais do IV SIC - Seminário de Iniciação Científica da UNIJUÍ, no campus Santa Rosa (RS), em 10/10/96, na página 203. Este trabalho apresenta o projeto formal de uma plataforma genérica para gerenciamento de redes heterogêneas. Com o objetivo de resolver problemas associados à distribuição de processos e suprir as necessidades das empresas de telecomunicações (associadas ao desenvolvimento eficiente de aplicações de gerência), a plataforma é ali definida para suportar aplicações de gerência. Especificar esta plataforma formalmente contribui no sentido de prover uma rigorosa descrição e validação, assim colaborando para sua precisão, especialmente nos estudos para atestar a conformidade entre o produto obtido e a descrição dos requisitos. Apresentado por Elenirse Furlanetto.

"Engenharia de Protocolos com o uso de Construções Predefinidas em LOTOS" é o título do artigo publicado nos anais do 2do Congresso Argentino en Ciencias de la Computacion - CACIC, nas páginas 451 à 462, de autoria de Cristiano Maciel, Mirella S.M.A. Notare e Bernardo G. Riso. O evento foi promovido pela Universidad Nacional de San Luis - UNSL, realizado na cidade de San Luis, na Argentina, no período de 7 a 11/12/1996. Neste trabalho apresenta-se o projeto da biblioteca bibLOTOS que armazena um conjunto de construções predefinidas

realizadas com a Técnica de Descrição Formal LOTOS. A bibLOTOS inclui construções em LOTOS Básico (onde apenas os aspectos de comportamento são definidos) e construções em LOTOS Completo (onde, além dos aspectos de comportamento, são considerados os aspectos de dados). A bibLOTOS é implementada utilizando o Gerenciador de Banco de Dados Access para microcomputadores. Um exemplo de utilização da bibLOTOS, no caso de um projeto de um sistema de gerência proativa de redes, é apresentado. Este evento foi importante pois permitiu estabelecer contatos com a comunidade latino-americana de computação e trocar informações acerca dos trabalhos em desenvolvimento. A apresentação do referido trabalho despertou o interesse dos congressistas, havendo busca de informações sobre a área de especificação formal e engenharia de protocolos. Apresentado por Cristiano Maciel.

O artigo "Especificação LOTOS de objetos gerenciados", de autoria de Ketter Rogerio Ohnes, Cristiano Maciel e Bernardo G. Riso foi publicado nos anais em CD-ROM do III INFTEL - Congresso PETROBRÁS de Informática e Telecomunicações, realizado no Rio de Janeiro (RJ), no período de 2 a 6/12/1996. Neste trabalho discute-se a utilização da TDF LOTOS para o projeto de objetos gerenciados, utilizados em sistemas de gerência de redes de computadores e redes de telecomunicações. Normalmente, tais objetos gerenciados são descritos com a utilização de recursos do GDMO (*Guidelines for the Definition of Managed Objects*) definidos a partir da linguagem ASN.1 (*Abstract Syntax Notation One*). Entretanto, as linguagens GDMO e ASN.1, além de não terem semânticas definidas formalmente, não dispõem de recursos para a representação de aspectos de comportamento de sistemas. Convém, portanto, investigar a possibilidade de utilização da TDF LOTOS para a representação e o projeto de objetos gerenciados na administração de redes. Apresentado por Cristiano Maciel.

Nos anais da III^{as} Jornadas de Informática e Investigación Operativa e V^{io} Encuentro del Laboratório de Ciência de la Computacion (anais ainda não disponíveis), realizado em Montevideo - Uruguay, deverá ser publicado o artigo intitulado "Biblioteca de Construções Predefinidas em LOTOS - a bibLOTOS", de autoria de Cristiano Maciel, Mirela S.M.A. Notare e Bernardo G. Riso. Este trabalho apresenta uma ferramenta para PC destinada aos usuários da Técnica de Descrição Formal LOTOS. Esta ferramenta, denominada bibLOTOS, é uma biblioteca que armazena construções predefinidas LOTOS. Estas construções predefinidas constituem um conjunto de especificações LOTOS, armazenadas e disponíveis para reutilização em um banco de dados, úteis no momento em que o projetista está desenvolvendo um sistema. Cada construção predefinida armazenada é acompanhada de outras informações, como por exemplo, o Sistema de Transições Rotuladas e uma implementação de referência em código C. A presente biblioteca é implementada no Sistema Gerenciador de Banco de Dados ACCESS 2.0©. Considera-se que uma vez desenvolvidos métodos efetivos e ferramentas adequadas de auxílio ao projetista torna-se possível a transferência da tecnologia LOTOS do ambiente acadêmico e de pesquisa para o ambiente industrial. Apresentado por Elenirse Furlanetto.

Nos anais do evento citado acima deverá ser publicado também o artigo "Desenvolvimento Formal de Sistemas para Gerência de Redes", de autoria de Elenirse Furlanetto e tendo como co-autores Cristiano Maciel, Mirela S.M.A. Notare e Bernardo G. Riso (anais ainda não disponíveis). Este trabalho apresenta alguns aspectos do desenvolvimento de sistemas de gerência de redes com emprego da TDF LOTOS. Tais sistemas têm sido desenvolvidos no Laboratório de Redes e Gerência seguindo uma estratégia de projeto baseada em refinamentos sucessivos, que é apoiada por ferramentas que agilizam e dão segurança ao trabalho dos

projetistas envolvidos. Para ilustrar o uso de tal estratégia, dois projetos são apresentados: uma plataforma para gerência de redes heterogêneas e um sistema de gerência proativa. Apresentado por Elenirse Furlanetto.

O artigo intitulado "Projeto Formal de una Plataforma para Gerenciamiento de Redes de Telecomunicaciones", de autoria de Mirela S.M.A Notare, Cristiano Maciel, Elenirse Furlanetto, Bernardo G. Riso e Carlos B. Westphall foi aceito para publicação e aguarda apresentação (16 e 17/04/97) no 3° ICIE97, na Universidad de Buenos Aires. O projeto formal de uma plataforma genérica para gerenciamento de redes heterogêneas tem como objetivo resolver problemas associados à distribuição de processos de gerência e suprir as necessidades das empresas de telecomunicações (no que se refere ao desenvolvimento de aplicações de gerência). Especificar esta plataforma formalmente contribui no sentido de prover uma rigorosa descrição e validação, assim colaborando para a sua precisão, especialmente nos estudos para atestar a conformidade entre o produto obtido e a descrição dos requisitos.

1.3 Organização desse trabalho

Este item tem como finalidade apresentar a estrutura do trabalho. Após esta *Introdução* (Capítulo 1) seguem mais oito capítulos, numerados de 2 até 9.

No Capítulo 2, intitulado *Revisão Bibliográfica*, faz-se um comentário sobre a literatura situada no campo da definição e utilização de construções predefinidas em LOTOS, bem como de métodos que fazem uso de Técnicas dc Descrição Formal na área de Gerência de Redes.

No Capítulo 3, intitulado *Construções Predefinidas*, define-se um conjunto de construções predefinidas geradas de acordo com um procedimento sistemático e classificadas segundo um código estruturado.

No Capítulo 4, intitulado *A Biblioteca bibLOTOS*, a concepção da biblioteca bibLOTOS é apresentada. Aspectos de sua implementação em ACCESS, as telas do sistema e a sua inserção no ambiente LOWE também são apresentados.

No Capítulo 5, intitulado *Método Proposto para o Desenvolvimento de Sistemas*, apresenta-se o método associado ao uso das construções predefinidas e da biblioteca implementada bibLOTOS.

No Capítulo 6, intitulado *Exemplo de Aplicação do Uso da bibLOTOS e do Método Associado*, um exemplo ilustrativo da aplicação de construções predefinidas para o projeto de um sistema de gerência de redes é desenvolvido.

Finalmente são apresentadas as *Conclusões e Propostas de Novos Trabalhos* (Capítulo 7) e as *Referências Bibliográficas* utilizadas como apoio à elaboração desse trabalho (Capítulo 8).

2. Revisão Bibliográfica

"Although the formal methodists still have a way to go, the informalists have even farther to go - particularly for critical systems" [Neum 96].

Este capítulo tem a finalidade de fazer referência às principais publicações que contribuíram para o delineamento do presente trabalho. As publicações aqui mencionadas estendem-se tanto na área de gerência de redes (com ênfase em trabalhos que abordam o projeto formal de sistemas nessa área) bem como publicações ligadas à questão da reutilização de especificações formais (especialmente aquelas voltadas para o uso de construções predefinidas).

2.1 Bibliografia sobre Gerência de Redes

O constante aumento da complexidade das redes de computadores e de telecomunicações tem estimulado a realização de estudos sobre as melhores técnicas para gerenciá-las [Schw 96]. Entre os objetivos destes estudos, destaca-se a definição de mecanismos para:

- controlar adequadamente o fluxo das informações que trafegam nas redes;
- coordenar a utilização dos recursos da rede, tornando mais eficiente essa utilização e melhorando a qualidade dos serviços fornecidos aos usuários;
- garantir a confiabilidade nas transmissões e solucionar os problemas que podem ocorrer periodicamente (por exemplo, problemas relacionados com a interoperabilidade de sistemas heterogêneos).

Em [Fran 96] são apresentadas considerações importantes sobre a gerência de redes. Nessa publicação afirma-se que os elementos da rede devem ser mantidos de modo a operar

eficientemente, permitindo a utilização de diferentes tipos de protocolos. Além disso, o funcionamento normal da rede deve ser mantido mesmo quando a população de usuários é crescente. Outro aspecto importante diz respeito ao projeto de sistemas de gerência. Estes projetos devem sempre estar atualizados, refletindo a real configuração do sistema implementado. Em [Fran 96] estuda-se a gerência proativa de redes e propõe-se um modelo para desenvolvimento de um sistema de gerência proativa que é composto por um Módulo de Gerência Proativa, uma Plataforma de Gerência e um Serviço de Comunicação. (No Capítulo 6 desse trabalho esse sistema de gerência proativa é retomado e especificado formalmente, em LOTOS, utilizando um conjunto de construções predefinidas).

No modelo de referência OSI (*Open Systems Interconnection*), definido pela ISO [ISO 7498.4], a gerência de redes está subdividida em cinco áreas funcionais: contabilização, segurança, falhas, configuração e desempenho. Esta subdivisão também é adotada pelo modelo de gerência da Internet.

A gerência de *contabilização* pode coletar e processar informações relacionadas ao consumo de recursos na rede [Rose 91]. Por outro lado, a gerência de *segurança* controla o acesso aos recursos da rede através de técnicas de autenticação e políticas de utilização e autorização (tipos de acesso) em uma rede. A gerência de *falhas* detecta anomalias no comportamento da rede, isola os problemas e tenta resolvê-los. Normalmente a gerência de falhas trata as ocorrências de eventos e notificações quando as anomalias surgem na rede.

A gerência de *configuração* destina-se a detectar e controlar o estado da rede (tanto a nível lógico quanto físico). A gerência de *desempenho* preocupa-se em controlar e analisar medidas de

performance da rede, como taxas de erros, vazão, tempo de resposta e taxa de utilização da rede. Estes dois últimos tipos de gerência (configuração e desempenho) podem ser associados, pois se o desempenho da rede não for satisfatório poderá haver necessidade de modificação da configuração da rede.

Nesse contexto, a gerência proativa vem sendo pesquisada. Entre os principais objetivos da gerência proativa, destacam-se [Jand 93]:

- manter o desempenho da rede em boas condições;
- ter disponível uma *baseline*, contendo indicações de dados anômalos ou críticos para distinguir o que é normal do que não é normal no funcionamento da rede;
- compreender diferentes padrões de utilização da rede, como por exemplo, recursos mais sobrecarregados, porcentagem de tráfego utilizando monitores de redes e analisadores de protocolos;
- verificar e determinar de que maneira a rede poderá suportar mais tráfego (melhorando o seu desempenho).

Em [Fran 96], [FrKo 96a] e [FrKo 96b] é proposto um modelo para o desenvolvimento da gerência proativa. Esse modelo é composto dos seguintes elementos:

1. O **Módulo de Gerência Proativa** é responsável por monitorar e analisar a rede, objetivando verificar as tendências de degradação da rede, conforme a aplicação especificada. Os componentes desse módulo são a *baseline*, um monitor Remoto e um serviço de verificação.

- a *Baseline* é um registro do comportamento normal da rede. Ela é freqüentemente consultada pelo Serviço de Verificação;

- o Monitor Remoto (ou Agente Remoto) coleta as amostras de objetos da MIB (*Management Information Base*) em tempo real;

- o Serviço de Verificação é responsável por examinar as tendências de degradação da rede. Esse exame é realizado por meio de comparações entre os valores armazenados na *Baseline* e os valores coletados, constantemente, pelo Monitor ou Agente Remoto;

2. A **Plataforma de Gerência** é responsável por receber as notificações e realizar as ações corretivas para impedir a degradação da rede. Normalmente as ações corretivas são determinadas pelo administrador da rede. No entanto têm sido propostos sistemas de correções automáticas com o uso de inteligência artificial [Roch 94][RoWe 96]. Para implementar um tal sistema em [RoWe 96] está sendo desenvolvida uma base de conhecimento com regras baseadas no comportamento da rede. O trabalho desenvolvido em [RoWe 96] é um dos resultados do Projeto PLAGERE ProTeM - CC - II/CNPq.

3. O **Serviço de Comunicação** é responsável por interconectar o Módulo de Gerência Proativa e a Plataforma de Gerência. O protocolo de gerência *Simple Network Management Protocol* (SNMP) foi utilizado pois a plataforma *SunNet Manager* tem um agente *proxy* SNMP e o agente remoto também é SNMP. Assim a troca de mensagens entre o Módulo de Gerência e a o Módulo Proativo Remoto é efetuada através do SNMP.

Uma representação gráfica do Modelo para a Gerência proativa de redes que segue a proposta apresentada em [DeKo 96a,b] e que está baseada na utilização do protocolo SNMP, pode ser vista na Figura 2.1.

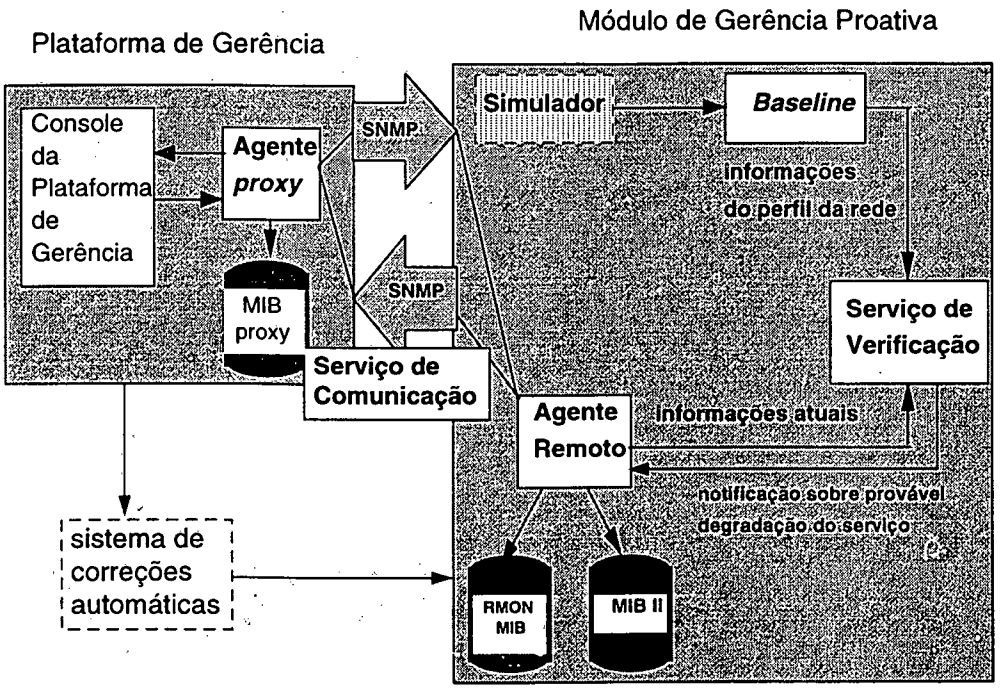


Figura 2.1 - Modelo para a gerência proativa de redes.

Em [MeNo 96] a TDF LOTOS é utilizada no projeto de agentes portáveis para gerência de redes de computadores. Neste trabalho inicialmente é apresentada uma visão geral sobre a área de gerência de redes e, posteriormente são ressaltadas as vantagens do uso de agentes portáveis.

Agentes Portáveis são usados no ambiente de gerenciamento de redes para buscar informações, criar, deletar e alterar o estado de objetos gerenciados. A denominação de agente portável dá-se pelo fato destes agentes poderem se adaptar facilmente a diferentes sistemas de gerenciamento [PeHo 95]. Um agente portável é especificado em LOTOS seguindo uma abordagem baseada em

refinamentos sucessivos. Em [MeNo 96] parte-se de uma especificação em alto nível de abstração e após a incorporação de sucessivas decisões de projeto obtém-se uma descrição detalhada do agente portável. De acordo com os autores de [MeNo 96], uma arquitetura genérica para agentes (baseada em seus componentes e funções básicas) facilita a criação e aumenta a funcionalidade dos agentes de gerenciamento. A abordagem proposta permite especificar de modo rigoroso os processos complexos que compõem a arquitetura de agentes portáveis.

Em [DeNo 95] apresenta-se o projeto de extensões à plataforma AIX NetView/6000 com a utilização de formalismos algébricos. A AIX NetView/6000 é uma plataforma de gerenciamento de redes para o ambiente System View da IBM e tem o objetivo de gerenciar redes baseadas no padrão TCP/IP através do protocolo SNMP.

Considerando que a gerência AIX NetView/6000 engloba apenas as áreas funcionais de configuração, falhas e desempenho, neste trabalho foi realizada uma extensão as aplicações do gerente para a área de segurança (segundo os autores, extensões na área de contabilização também poderiam ser desenvolvidas com a utilização do método apresentado nesse trabalho).

Uma aplicação possível dentro desta área de segurança é o controle de *log*. Tal controle pode ser realizado de forma que os agentes enviem *traps* ao gerente quando um limiar de tentativas de acesso frustadas for alcançado. Por outro lado, o gerente pode requisitar informações ao agente a qualquer momento. A comunicação entre o gerente e o(s) agente(s) é feita utilizando o protocolo SNMP, através das seguintes primitivas: *get_request, get_next_request, get_response, set_request, trap.*

Em [DeNo 95] especifica-se formalmente o comportamento do gerente (através de extensões), com o uso da TDF LOTOS. Parte-se de uma especificação formal em alto nível de abstração e então é feito o refinamento do comportamento do gerente *NetView*. Dentro das aplicações identificadas é especificada a aplicação correspondente *à* área de segurança.

As ferramentas *Cæsar, Cæsar.open, Aldébaran* e TOPO são utiiizadas para análise e validação das especificações. Tal trabalho pretendeu contribuir com um estudo para o projeto formal de novas aplicações para a gerência *NetView*.

Em [NoRi 95a] apresenta-se a descrição formal de uma plataforma para auxiliar a gerência de redes heterogêneas. O formalismo utilizado para especificar os aspectos de comportamento da plataforma é a TDF LOTOS Básico. Nesse trabalho inicialmente é especificado o sistema de gerência que reúne a plataforma, aplicações genéricas e aplicações comuns. Posteriormente é descrito o serviço oferecido pela plataforma e também o protocolo correspondente ao recurso de comunicação dessa plataforma.

Ao contrário das soluções de gerenciamento proprietárias, que são de difícil interoperabilidade, a plataforma proposta em [NoRi 95a] foi concebida para integrar aplicações genéricas de gerência. Essa plataforma possui duas interfaces. O primeiro tipo de interface (interface superior da plataforma) refere-se à interação da plataforma com as aplicações genéricas de gerenciamento, permitindo a comunicação da plataforma com os agentes e gerentes das aplicações genéricas de gerência. O segundo tipo de interface (interface inferior da plataforma) refere-se à interação da plataforma com as aplicações de uso comum da camada de aplicação, permitindo à plataforma o

uso do protocolo SNMP e do CMIS/CMIP, entre outros protocolos de comunicação para controlar o tráfego de informações de gerência de redes.

Em [NoRi 95a] a especificação da plataforma é realizada com a utilização da TDF LOTOS e de um método proposto previamente em [Nota 95]. Tal método é baseado em estilos de especificação, refinamentos sucessivos e transformações. Através do estilo orientado para restrições e do estilo orientado para recursos são descritos o serviço e o protocolo de comunicação da aplicação da área funcional de contabilização de gerência de LOG (para contabilizar quem usa os recursos da rede). Para a descrição do protocolo são utilizadas transformações sobre a especificação do serviço oferecido pela plataforma e do serviço subjacente à plataforma. Métodos baseados em transformações são também apresentados em [BoGo 86] e [Pire 94]. Nestas publicações são relatadas as dificuldades de desenvolvimento de trabalhos nessa área, principalmente aqueles relacionados às regras de transformações que não conseguem manipular mais do que exemplos triviais.

2.2 Bibliografia sobre Métodos de Projeto

Diferentes técnicas formais e diferentes métodos vêm sendo sugeridos para o projeto formal de sistemas de gerência de redes. Algumas publicações que envolvem a utilização de técnicas formais e métodos são apresentados sucintamente nesta seção.

Em [BoMo 91] apresenta-se um método de projeto baseado na orientação a objetos, fazendo uso de uma linguagem de especificação formal (a linguagem MONDEL) para o desenvolvimento de

sistemas para o gerenciamento de redes. Tal metodologia está baseada na representação dos objetos com o uso do conceito de entidade-relacionamento, muito aplicado ao projeto de *software*, onde as entidades correspondem aos arquivos e os relacionamentos correspondem à interligação dos aquivos através dos seus atributos. As vantagens desta abordagem de projeto são numerosas, embora esta permaneça muito informal e intuitiva, uma vez que o desenvolvimento dos processos baseia-se na perícia humana.

Os métodos para o desenvolvimento de sistemas de acordo com uma abordagem orientada a objetos freqüentemente estão organizadas em um certo número de passos sucessivos. Quatro passos são propostos em [BoMo 91]. O primeiro passo consiste na identificação do problema, de maneira informal. O segundo passo consiste em identificar os componentes chaves da assim chamada aplicação domínio, descrevendo as funcionalidades e produzindo um conjunto de entidades como resultado. No terceiro passo busca-se a alocação das funções apresentadas como operações pelos objetos identificados no passo anterior. O passo final é a definição do comportamento das várias entidades correspondentes aos objetos das operações alocadas. O sucesso deste passo depende, principalmente, do conhecimento e da experiência do projetista na área.

A linguagem de especificação orientada a objetos, MONDEL, desenvolvida por um grupo de pesquisa [BoBa 89] suporta o método descrito acima. De acordo com os autores esta linguagem descreve apropriadamente sistemas em tempo real.

Pode-se dizer que a linguagem MONDEL é compatível com várias notações usadas no contexto de padronização OSI, como, por exemplo, a notação ASN.1. A linguagem MONDEL é similar a linguagem ADA.

MONDEL possui um compilador que verifica a sintaxe e a semântica das especificações. Esse compilador também traduz a especificação para uma forma intermediária, a qual pode ser usada para execução em um interpretador escrito em *Prolog*. O trabalho com validação exaustiva das especificações MONDEL estão em andamento.

Em [KoBo 91] são apresentadas especificações de Modelos de Informação para Gerência usando Técnicas de Descrição Formal. Um ambiente com propósito de realizar atividades de gerência é proposto usando LOTOS. O uso de LOTOS permite definir o Modelo de Informação para Gerência sem ambigüidades. O uso de ferramentas para análise e verificação das especificações permite validar o projeto. No sistema proposto a comunicação gerente-agente é descrita formalmente. O comportamento do agente (visto pelo gerente) é especificado em LOTOS, de modo genérico, isto é, de modo que é necessário uma representação específica para cada sistema gerenciado particular. O agente é definido com a utilização de tipos abstratos de dados.

Em [Nota 95] apresenta-se um método para a especificação formal de serviços e protocolos de comunicação. Este método permite a obtenção de uma especificação de protocolo a partir de uma especificação de serviço correspondente e da especificação de serviço subjacente. Tal método respeita princípios de projeto (generalidade, flexibilidade e ortogonalidade), utiliza estilos de especificação (estilo monolítico, estilo orientado a restrições, estilo orientado para recursos e estilo orientado para estados) e segue uma abordagem de projeto baseada em refinamentos

sucessivos, que propõe o uso de construções predefinidas e transformações de especificações. Um método proposto é aplicado na especificação de uma plataforma de apoio à gerência de redes heterogêneas.

Em [Toni 96] é apresentado um estudo na área de engenharia de Protocolos de Comunicação para redes de computadores, redes de telecomunicações e sistemas distribuídos. Nesse contexto o estudo volta-se, principalmente, para as abordagens de projeto que visam a automatização do processo de síntese de protocolos. Tais abordagens baseiam-se em regras de transformação que, uma vez aplicadas à especificação de um serviço distribuído, geram, automaticamente, a especificação do protocolo correspondente a esse serviço [ToNo 95]. Um conjunto de algoritmos de transformação para a obtenção dos protocolos correspondentes são oferecidos e implementados computacionalmente. Esses algoritmos são aplicáveis a especificações de serviço realizadas com a Técnica de Descrição Formal LOTOS. Tal implementação constitui a ferramenta TranSP, a qual serve de auxílio à realização de projetos de Engenharia de Protocolos de Comunicação [Toni 95].

Em [MeNo 96] discute-se a utilização da TDF LOTOS no projeto de agentes portáveis para a gerência de redes de computadores. O trabalho inicia com uma visão geral sobre a gerência de redes de computadores e discute as vantagens do uso de Agentes Portáveis. Um Agente Portável é especificado em LOTOS. O uso da Técnica de Descrição Formal na especificação do Agente Portável contribui para a definição de uma arquitetura genérica para agentes, baseada em seus componentes e operações básicas, facilitando a criação e aumentando a funcionalidade dos agentes de gerenciamento.

2.3 Bibliografia sobre Reutilização de Especificações

A reutilização de especificações (e de seus componentes) pode ser uma promissora abordagem para o desenvolvimento de métodos de projeto que proponham o uso de técnicas formais. Nesta seção busca-se apresentar alguns elementos de abordagens que se apóiam na reutilização de especificações formais.

2.3.1 Agentes Básicos

Em [Miln 80] são apresentadas agentes básicos em CCS (*Calculus Of Communicating Systems*). Tais agentes básicos, em alguns casos, incluem a definição de dados (de modo informal).

São agentes básicos predefinidos sem passagem de valores [Miln 80]**:**

`Ciclo` - é um agente que repete infinitamente uma seqüência de três eventos. Veja a Figura 2.2.

```
CICLO = a.b.c.CICLO
```

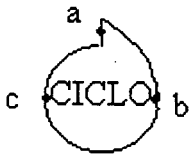


Figura 2.2 - Processo `CICLO`.

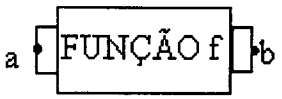
`Disjuntor` - é um agente que, após um evento na porta (a) fica sujeito a uma escolha indeterminística com (duas) alternativas. Em cada alternativa há um retorno recursivo. Veja a Figura 2.3.

```
DISJUNTOR = a . (b . DISJUNTOR
                +
                c . DISJUNTOR)
```

Figura 2.3 - Processo DISJUNTOR.

São agentes básicos predefinidos com passagem de valores [Miln 80]:

`Função f` - é um agente que executa uma função unária simples. Veja a Figura 2.4.




```
FUNÇÃO f = ax.bf(x). FUNÇÃO f
```

Figura 2.4 - Processo `FUNÇÃO f`.

`PREDICADO p` - é um agente predicado unário. Veja a Figura 2.5.


```
PREDICADO  p = ax.if p(x) then bx.PREDICADO p
                          else cx.PREDICADO p
```

Figura 2.5 - Processo `PREDICADO p`.

Ainda outros agentes básicos são encontrados em [Miln 80].

2.3.2 As construções PDICs

Em [PiSi 92] introduz-se o conceito de PDICs (Implementações de Construções Predefinidas) em LOTOS como um recurso para o projeto de sistemas distribuídos. Ali discute-se como PDICs podem ser selecionadas e usadas, e ilustra-se a aplicabilidade deste recurso usando-o em um exemplo de projeto: a implementação de um protocolo de janela deslizante. Segundo os autores, métodos automatizados para transformar especificações genéricas numa composição de PDICs ainda não estão disponíveis, sendo esta uma área interessante para pesquisa. O autor expõe que o uso de construções predefinidas faz clara distinção entre projetos estruturais (ligados à arquitetura dos serviços) e projetos tecnicamente inerentes aos diferentes ambientes de implementação (ligados ao protocolo e seu detalhamento).

São exemplos da biblioteca de PDICs apresentada [PiSi 92]**:**

Construção `Do_f`. Esta construção recebe alguns valores de dados como entrada, e entrega o resultado da aplicação da função `f` sobre os valores de entrada como saída. Veja a Figura 2.6.

```
process Do_f [Inp,Out] : noexit :=
   inp ?x : Element ;
   ([x IsIn Domain_f]->out !f(x);Do_f Inp,out]
    [][x IsNotIn Domain_f] -> Do_f [Inp,Out])
endproc (* Do_f *)
```

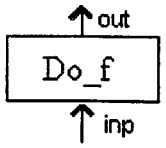


Figura 2.6 - Construção `Do_f`.

Construção `Choice_p`. Esta construção representa um comportamento no qual um dado é admitido como entrada, e um predicado aplicado sobre esse dado decide entre duas alternativas possíveis. Veja a Figura 2.7.

```
process Choice_p [inp, out1, out2] :noexit :=
   inp ?x : Element;
   ([p(x)]->out1 ; Choice_p[inp,out1,out2]
   [][not(p(x))]->out2;Choice_p[inp,out1,out2])
endproc (* Choice_p *)
```

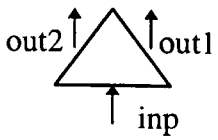


Figura 2.7 - Construção `Choice_p`.

2.3.3 Outras construções predefinidas

Em [Riso 93] realiza-se um estudo visando a produção de construções predefinidas de maneira sistemática assim como a utilização dessas construções no desenvolvimento de especificações. Nesse caso é empregada a TDF LOTOS Básico e vários exemplos simples são apresentados.

Dentre as construções predefinidas apresentadas em [Riso 93] podem-se citar:

Construção Predefinida GERADOR. Neste processo ocorre a emissão de um valor, seguido do processo básico **stop**.

```
process GERADOR[sai-valor] : noexit : =
   sai-valor; stop
endproc
```

Construção Predefinida COMUTADOR_LIGADO. Este processo representa um comutador que funciona apenas uma vez.

```
process COMUTADOR_LIGADO[liga,entrada,estimulo-1,saida-1,estimulo-2,saida-2] : noexit: =
   liga;entrada;(estimulo-1;saida-1;stop
                              [ ]
                 estimulo-2,saida-2,stop)
endproc
```

Construção Predefinida CONTFIM. O processo CONTFIM pode ser obtido através da rerrotulação do processo básico COMUTADOR-LIGADO, apresentado acima.

```
Process CONTFIM [cont,u-req,u-pos,u-neg,pos,neg-fim] : noexit: =
   cont;u-req;(u-pos;pos;stop
                          [ ]
               u-neg;neg-fim;stop)
endproc
```

Por sua vez, em [NoRi 94 a] apresenta-se um estudo preliminar sobre construções predefinidas buscando incluir estas construções em um processo de transformação automática de especificações. Nesse trabalho é apresentado um sistema do tipo Pergunta-Resposta é apresentado como exemplo.

Algumas construções predefinidas são abordadas nesse trabalho, a saber:

Construção Predefinida `le_escreve`. Neste processo a escrita é habilitada pela leitura. Pode-se observar que no processo `le_escreve` os subprocessos `leitura[in]` e `escrita[out]` são muito simples e podem ser considerados, eles mesmos, como construções predefinidas.

```
process le_escreve[in, out]: noexit :=
   leitura[in] >> escrita[out]
where
   process leitura[ in ]: exit :=
      in; exit
   endproc
   process escrita[ out ]: noexit :=
      out; stop
   endproc
endproc
```

Construção Predefinida `Multi_recursivo`. Em [NoRi 94 a] são também definidos sistemas onde a recursividade não é usada em todas as opções. No exemplo a seguir, o primeiro somatório corresponde às opções que terminam em inatividade, enquanto que o segundo somatório corresponde às opções com recursividade. Veja a Figura 2.8.

process Multi_recursivo[$a_1,\ldots,a_n$]**:noexit :=**

$$\sum_{i=1}^{m}(a_i;stop)$$

[ ]

$$\sum_{i=m+1}^{n}(a_i;Multi\_recursivo[a_{m+1},\ldots,a_n]$$

endproc

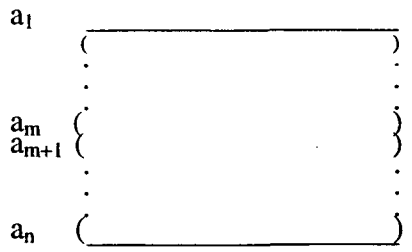


Figura 2.8 - Processo `Multi_recursivo`.

Situações mais complexas envolvem escolhas com seqüências de ações (repetidas ou não) onde pode haver a intercalação de eventos observáveis com eventos não-observáveis (este tipo de evento é representado pela letra `i`). Cada escolha pode desdobrar-se em várias novas opções.

Em [Nota 95] sugere-se o emprego de construções predefinidas em um projeto que tem como escopo o gerenciamento de redes, através da utilização de uma biblioteca que armazena estas construções. Em [Nota 95] foram utilizadas algumas construções predefinidas no desenvolvimento de um sistema de gerência que engloba uma plataforma para redes heterogêneas.

Estratégias de projeto e implementação de sistemas são discutidas em [BoVa 95]. O uso de PDICs (Implementações de Construções Predefinidas) em LOTOS é abordado e aplicado no projeto de um sistema designado como Serviço de Transferência de Dados (DTS). A arquitetura inicial do DTS é apresentada informalmente e a estrutura da implementação correspondente é representada em LOTOS. A implementação é traduzida para uma composição de PDICs, de acordo com a abordagem proposta pelos autores de [BoVa 95]. O primeiro passo da abordagem é a definição da biblioteca de PDICs. Uma biblioteca simples é apresentada.

Veja abaixo algumas PDICs apresentadas nesse trabalho:

Construção Predefinida `Timer`. Esta construção dá ênfase às informações de tempo para ambiente multi-tarefas. Estas condições de tempo não podem ser formalmente representadas em LOTOS, sendo que a definição desta construção é de grande importância, principalmente no controle de fluxo, onde ocorrem eventos em paralelo de forma independente. Os aspectos quantitativos (valores do tempo trancorrido) podem ser descritos informalmente até que a

construção Timer seja implementada. Uma construção Timer pode ser especificada em LOTOS como segue:

```
process Timer [set, expired, reset] : noexit :=
     set; ( i ;expired ; exit [] reset ; exit)
     >> Timer [set , expired , reset]
endproc
```

Na especificação acima o evento set ativa o temporizador. Após a ativação pode ocorrer o evento interno i, seguido do evento expired, indicando o estouro da temporização. Após esse estouro o Timer pode ser reinicializado. Alternativamente, após a ativação, e antes do estouro da temporização, o Timer pode ser reinicializado com a ocorrência do evento reset.

De acordo com os autores, esta construção pode ser utilizada no projeto de sistemas operacionais e implementada como uma tarefa em ambientes multi-tarefas.

Construção Predefinida Scheduler. Esta construção recebe alguns dados de entrada e distribui estes dados para outros componentes. Muitos exemplos de *Scheduler* podem ser definidos devido a alteração do número de interações, dos valores recebidos e transmitidos, etc. Aqui são apresentadas três interações com o *Scheduler* que permitem receber um valor e distribuir este valor para outros dois processos através de interações ordenadas. Veja abaixo uma construção *Schuduler* como um processo LOTOS.

```
process Scheduler [inp, out1, out2] : noexit :=
inp ?x : Element; out1 !x; out2 !x ; Scheduler [inp, out1, out2]
endproc
```

Schedulers podem ser implementados como tarefas em ambientes multi-tarefas e utilizados na implementação de alguns casos especiais de sincronização multi-vias, no qual os processos envolvidos não envolvem restrições em interações e o *scheduler* pode ser considerado seguro.

A revisão bibliográfica apresentada neste capítulo busca esboçar um panorama da pesquisa com relação a gerência de redes e métodos de projeto, sobretudo métodos que abordam a reutilização de especificações. Assim alguns métodos de projeto que fazem uso de TDFs foram estudados para a obtenção de subsídios que permitissem a elaboração do método proposto no Capítulo 5 (Método Proposto para o Desenvolvimento de Sistemas). Na evolução dos estudos sobre a reutilização de especificações nota-se que o conceito de **agentes básicos** em CCS, introduzido por Milner [Miln 80], teve continuidade, anos mais tarde, com o surgimento das **Implementações de Construções Predefinidas** (PDICs) em LOTOS, proposto por [PiSi 92]. Em [Riso 93] busca-se sistematizar a definição de **construções predefinidas** em LOTOS. Em [NoRi 94a] apresenta-se um estudo preliminar sobre **construções predefinidas** e em [Nota 95] o uso de construções prefinidas com o método de projeto é proposto, bem como o armazenamento destas em uma biblioteca.

O presente levantamento bibliográfico permite concluir que há uma tendência para o uso de construções predefinidas que pode ser de utilidade para a construção de uma biblioteca estruturada que armazene um conjunto de construções predefinidas LOTOS. O uso de tais construções deve estar baseada em um método de projeto que oriente o uso dos recursos oferecidos por essa biblioteca.

3. Construções Predefinidas

"Each method starts with a set of inputs that are grouped to support the technical analysis that is performed" [Evan 89].

O conceito de construções predefinidas, cuja evolução foi investigada no Capítulo 2 (Revisão Bibliográfica), é objeto de estudo neste capítulo. Agora o objetivo é aprensentá-las de modo organizado, classificando-as e armazenando-as em uma biblioteca apropriada.

No contexto deste trabalho as construções predefinidas são especificações LOTOS. O esforço para a concepção de construções predefinidas orienta-se para a produção de construções LOTOS de uso geral. Elas são previamente armazenadas em uma biblioteca (a bibLOTOS), e posteriormente utilizadas à medida que o projetista está descrevendo e analisando um dado sistema.

A correção das construções predefinidas já está estabelecida antes destas serem usadas em algum projeto específico, de modo que a correção da especificação final de um sistema torna-se exclusivamente dependente da correção da montagem realizada pelo projetista [BoLa 95].

3.1 Vantagens do uso de construções predefinidas

As construções predefinidas podem ser úteis em todas as fases do ciclo de vida de um sistema, isto é, convém empregá-las tanto na descrição quanto na análise, verificação, simulação, teste e manutenção de sistemas.

A **descrição** de um sistema com o uso de construções predefinidas permite ganhar em produtividade, pois não é necessário reescrever uma construção já editada uma vez. Ela é simplesmente transportada da bibLOTOS para a especificação em curso.

A fase de **análise** é beneficiada, por exemplo, através da visualização da representação gráfica correspondente a uma construção predefinida, e também pela facilidade de exame do sistema de transições rotuladas (LTS) correspondente.

A **validação** de construções predefinidas (através de simulações, testes e verificações) pode ser dispensada, uma vez que as construções predefinidas só devem ser armazenadas após validadas. Além disso as construções predefinidas são úteis tanto na **simulação** como nos **testes** e **verificações** de sistemas, uma vez que elas são facilmente inseridas na especificação e dali removidas, para cada tipo de validação desejada. O processo de validação de especificações utiliza principalmente os Sistemas de Transições Rotuladas (LTSs) da bibLOTOS.

Cada construção predefinida pode ser associada a diferentes tipos de dados e mapeada para construções em linguagens de programação, por exemplo, linguagem C, facilitando as tarefas de **implementação** e **manutenção** de sistemas. Uma vez que na biblioteca bibLOTOS tem-se o código C correspondente a cada construção predefinida, a manutenção de sistemas pode ser beneficiada pelo uso de especificações já traduzidas (e validadas) para linguagens de implementação (no caso, linguagem C).

3.2 Classificação das construções predefinidas

Antes de serem reutilizadas, as construções predefinidas podem ser elaboradas, classificadas e armazenadas em um repositório. Elas podem ser classificadas de acordo com as características de seus comportamentos ou pelo uso ou não de estruturas de dados, ou pelo uso de ambos, facilitando assim a busca e a reutilização no sistema em desenvolvimento.

Na concepção do acervo disponível na bibLOTOS é utilizado, inicialmente, um procedimento para a geração sistemática de construções predefinidas que parte dos casos mais simples (processos básicos como **stop** e **exit**) e evolui para construções mais complexas (comportamentos finitos, infinitos e comportamentos indeterminísticos), chegando a construções com maior grau de funcionalidade (como, por exemplo, as construções usadas no escopo de sistemas de gerência de redes), estruturas de dados, ou ambos. [MaNo 96b].

As construções predefinidas presentes na bibLOTOS, apresentadas na seção 3.3 deste capítulo, são nomeadas através de um código estruturado e separadas em três grandes grupos. O primeiro grupo classifica as construções predefinidas de acordo com o comportamento dos processos definidos em LOTOS Básico (como por exemplo, comportamentos finitos ou infinitos; determinísticos ou indeterminísticos; eventos observáveis ou internos). O segundo grupo é classificado pelo uso tipos abstratos de dados em ACT ONE. No terceiro grupo estão as construções predefinidas que representam comportamentos e dados em LOTOS Completo. Veja a Figura 3.1.

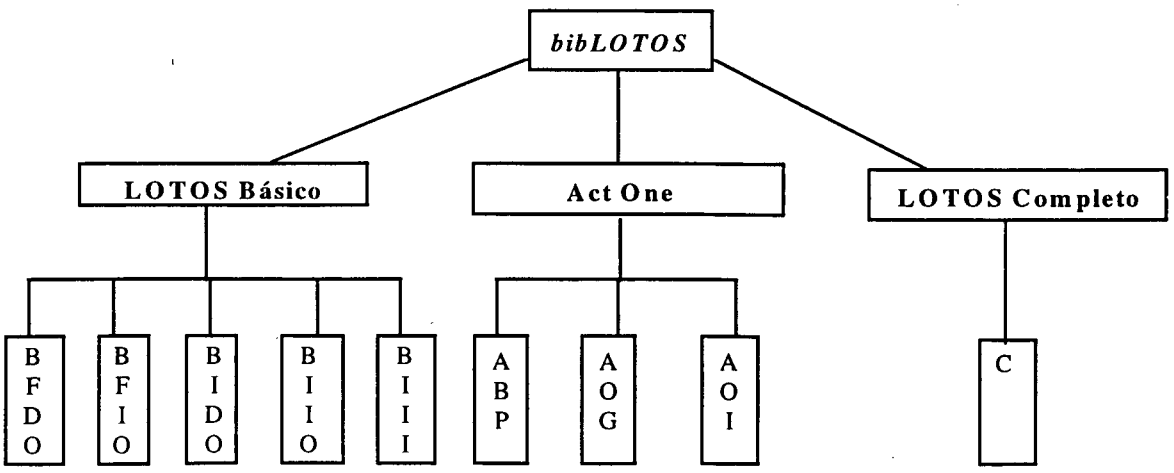


Figura 3.1 - Representação em estrutura de árvore da classificação utilizada para identificar as construções predefinidas da bibLOTOS.

O campo código da bibLOTOS armazena o código estruturado da construção predefinida de acordo com a classificação descrita nos itens 3.2.1 (LOTOS Básico), 3.2.2 (ACT ONE) e 3.2.3 (LOTOS Completo).

3.2.1 LOTOS Básico

Os processos básicos (**stop** e **exit**) e operadores básicos (; [ ], ||| , || , |[ ]|, etc) da linguagem LOTOS permitem a representação dos aspectos de comportamento dos processos e as interações a que estão sujeitos. Usando tais processos e operadores pode-se criar construções predefinidas. Veja a Figura 3.2.

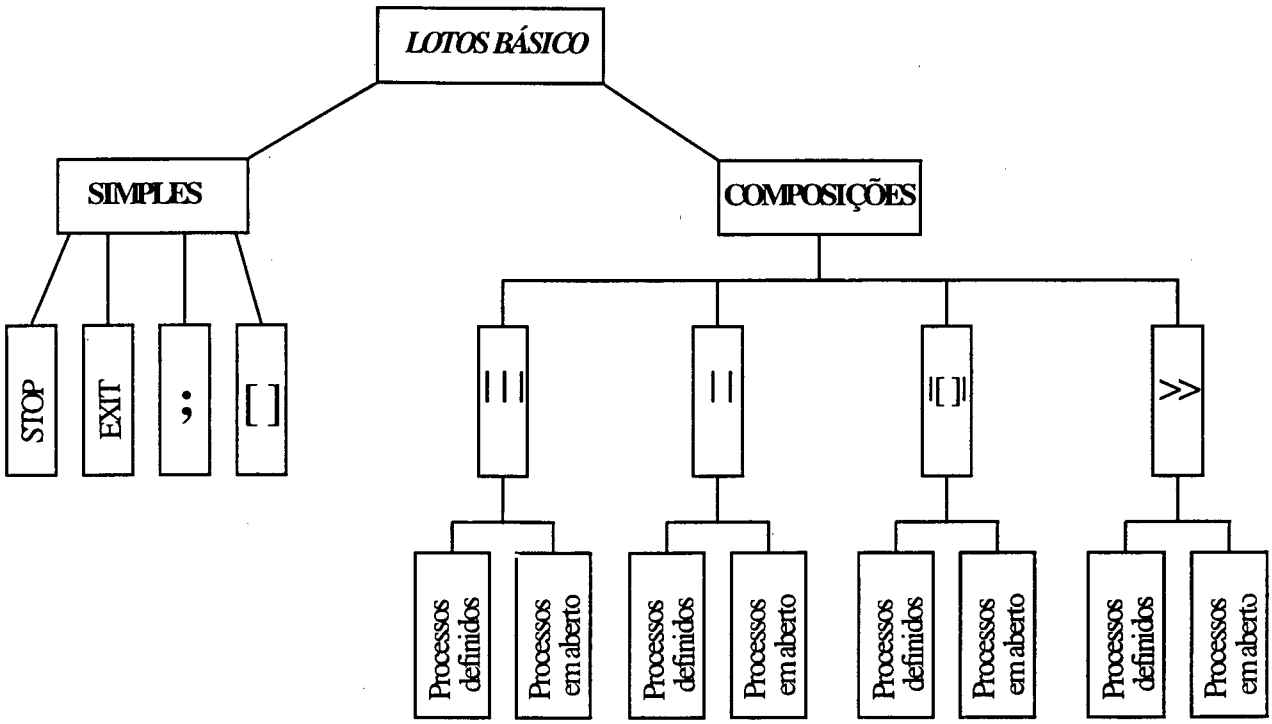


Figura 3.2 - Representação em estrutura de árvore das construções predefinidas LOTOS Básico.

Quanto à representação do comportamento, os processos podem ser classificados em:

- processos finitos (que, eventualmente, terminam) ou processos com possibilidades infinitas (que são especificados com o emprego da recursividade);

- processos determinísticos (processos que não apresentam escolhas entre comportamentos) ou processos indeterminísticos (processos que apresentam escolhas indeterminísticas entre expressões de comportamento);

- processo com somente eventos observáveis (não contém nenhum evento interno `i`) ou processos com eventos internos `i`.

Esta classificação de comportamento dos processos permite identificar as construções predefinidas em LOTOS Básico utilizando quatro caracteres. Esses *caracteres* formam o código estruturado para identificação da construção predefinida (BFDO, BFIO, BIDO, BIIO ou BIII), seguidos do nome do processo, conforme apresentado a seguir:

- 1° caráter = LOTOS **B**ásico;

- 2° caráter = processo **F**inito/processo com possibilidade **I**nfinita;

- 3° caráter = processo **D**eterminístico/**I**ndeterminístico;

- 4° caráter = processo com somente eventos **O**bserváveis/processo com eventos **I**nternos;

- 5° caráter = caráter sublinha (_) como separador do código estruturado e do nome do processo.

- Demais caracteres = nome do processo definido pelo projetista.

Veja este simples exemplo: **process** BFDO_SEQ1[a1] : **noexit** : =

a1; **stop**

endproc

Nesse caso trata-se de uma construção predefinida em LOTOS Básico (**B**), com comportamento finito (**F**), determinístico (**D**) e sem eventos internos (**O**). O processo sugere uma seqüência de eventos, nomeado por **SEQ1**. O código desta construção predefinida então é **BFDO_SEQ1**.

3.2.2 ACT ONE

O tratamento dos dados na especificação formal de sistemas baseia-se na teoria de tipos abstratos de dados. Com o uso de ACT ONE [EhMa 85] podem ser especificados os aspectos de dados de protocolos de comunicação em alto nível de abstração, o que implica na descrição desses dados independentemente do ambiente de implementação. ACT ONE é uma técnica que dispões de rigoroso embasamento matemático.

O fato da teoria de tipos abstratos de dados ter como base um modelo matemático permite que o sistema especificado seja validado por meio de testes, simulações e verificações. No entanto, os tipos abstratos de dados nem sempre são empregados, possivelmente, devido à sua complexidade. Existem alguns casos de especificação de tipos de dados em que o projetista demanda muito tempo especificando tipos de dados elementares, o que é um dos principais obstáculos ao uso deste recurso de projeto [TeCh 96].

A realização de aplicações práticas das técnicas de descrição formal fica facilitada se métodos de suporte ao projetista forem disponibilizados. No caso de especificações ACT ONE é de interesse a construção de bibliotecas que armazenem construções predefinidas reutilizáveis.

A proposta de uma ferramenta de suporte para permitir a reutilização de construções predefinidas ACT ONE é aqui apresentado. Esta ferramenta tem aplicação em diversas áreas, em especial, na área de redes de computadores e telecomunicações. O método proposto oferece a possibilidade de reutilizar uma construção predefinida ACT ONE com semântica similar à que o projetista necessita, proporcionando ao projetista a liberdade de alterá-la, obtendo uma nova especificação.

Nesta dissertação as construções predefinidas ACT ONE são classificadas em:

- ACT ONE Biblioteca Padrão (ABP): constituída pelos tipos abstratos de dados elementares apresentados em [ISO 8807];

- ACT ONE Objetos Gerenciados (AOG): construções predefinidas que representam estruturas de dados de objetos gerenciados utilizados em sistemas de gerência de redes de computadores e telecomunicações;

- ACT ONE Outros Tipos de Dados (AOTD): construções predefinidas que representam outros tipos de dados úteis na especificação de diversos sistemas, por exemplo, filas, pilhas e outros tipos de dados). Veja a Figura 3.3.

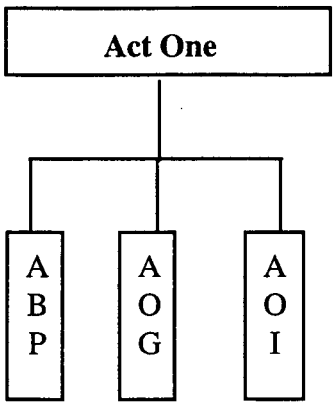


Figura 3.3 - Representação em estrutura de árvore das construções predefinidas ACT ONE.

De acordo com esta classificação, as contruções predefinidas ACT ONE são identificadas pelos *caracteres* que formam o código estruturado, conforme apresentado a seguir:

- 1° caráter = **ACT** ONE;

- 2° e 3° caráter = **Bi**blioteca **Pa**drão;

- 4° caráter = caráter sublinha (_) como separador do código estruturado e do nome do processo.

- Demais caracteres = nome do processo definido pelo projetista.

3.2.3 LOTOS Completo

As contruções predefinidas especificadas em LOTOS Completo representam comportamentos (em LOTOS Básico) e estruturas de dados (em ACT ONE).

O produto cartesiano das construções predefinidas LOTOS Básico com as construções predefinidas ACT ONE dão origem a uma grande quantidade de construções predefinidas descritas em LOTOS Completo. Veja a Figura 3.4.

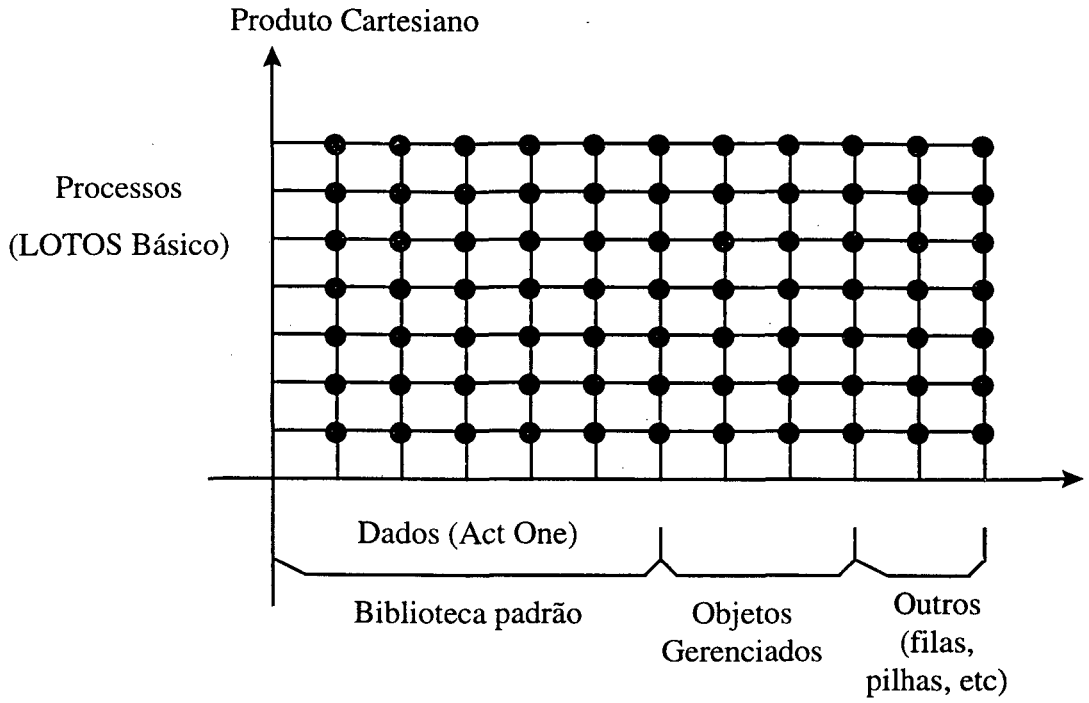


Figura 3.4 - Construções predefinidas em LOTOS Completo.

As contruções predefinidas LOTOS Completo são identificadas pelos *caracteres* que formam o código estruturado (C), conforme apresentado a seguir:

- 1° caráter = LOTOS Completo;

- 2° caráter = caráter sublinha (_) como separador do código estruturado e do nome do processo.

- Demais caracteres = nome do processo definido pelo projetista.

3.3 Construções predefinidas da bibLOTOS

As especificações abaixo relacionadas já fazem parte da biblioteca de construções predefinidas bibLOTOS. Muitos outros exemplos estão sendo incorporados a esta biblioteca, em diversos estilos de especificação. As construções predefinidas estão agrupadas conforme as suas características principais. Tais características correspondem a um código de classificação que reúne as construções predefinidas em três segmentos principais, a saber: construções predefinidas LOTOS Básico, construções predefinidas ACT ONE e construções predefinidas LOTOS Completo.

3.3.1 Construções Predefinidas em LOTOS Básico

As construções predefinidas em LOTOS Básico permitem definir a arquitetura e o comportamento dos sistemas.

3.3.1.1 Construções Predefinidas em LOTOS Básico, processos Finitos, Determinísticos e com somente eventos Observáveis (BFDO)

Nesta seção são apresentadas vinte e oito construções predefinidas em LOTOS Básico, cujos processos são finitos, determinísticos e com somente eventos observáveis. Estas construções predefinidas são classificadas como BFDO.

```
(01)   process BFDO_SEQ1[a1] : noexit : =
           a1; stop
       endproc
```

```
(02)   process BFDO_SEQ2[a1] : noexit : =
           a1; p;
           where
              process p ... endproc
       endproc
```

```
(03)  process BFDO_SEQ3[a1] : noexit : =
          a1;BFDO_INTERRUPCAO1[interrompe,motivo]
          where
              process BFDO_INTERRUPCAO1[interrompe,motivo] ... endproc
      endproc
```

```
(04)  process BFDO_SEQ4[a1] : noexit : =
          a1;BFDO_INTERRUPCAO2[interrompe,motivo]
          where
              process BFDO_INTERRUPCAO2[interrompe,motivo] ... endproc
      endproc
```

```
(05)  process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : =
          interrompe;motivo;stop
      endproc
```

```
(06)  process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : =
          interrompe;motivo; p;
          where
              process p ... endproc
      endproc
```

```
(07)  process BFDO_GERADOR1[sai-valor] : noexit : =
          sai-valor;stop
      endproc
```

```
(08)  process BFDO_GERADOR2[sai-valor] : noexit : =
          sai-valor; p;
          where
              process p ... endproc
      endproc
```

```
(09)  process BFDO_GERADOR3[sai-valor] : noexit : =
          sai-valor; BFDO_INTERRUPCAO1[interrompe,motivo]
          where
              process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : =
                  interrompe;motivo;stop
      endproc
```

```
(10)  process BFDO_GERADOR4[sai-valor] : noexit : =
        sai-valor; BFDO_INTERRUPCAO2[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : =
              interrompe;motivo;p;
              where
                  process p ... endproc
            endproc
      endproc
```

```
(11)  process BFDO_SEQUENCIA1[liga,a,b,c] : noexit : =
        liga;a;b;c;stop
      endproc
```

```
(12)  process BFDO_SEQUENCIA2[liga,a,b,c] : noexit : =
        liga;a;b;c;p
        where
            process p ... endproc
      endproc
```

```
(13)  process BFDO_SEQUENCIA3[liga,a,b,c] : noexit : =
        liga;a;b;c; BFDO_INTERRUPCAO1[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : =
                interrompe;motivo;stop
      endproc
```

```
(14)  process BFDO_SEQUENCIA4[liga,a,b,c] : noexit : =
        liga;a;b;c; BFDO_INTERRUPCAO2[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : =
                interrompe;motivo;p;
                where
                    process p ... endproc
      endproc
```

```
(15)  process BFDO_FUNCAO_BINARIA1[entrada1,entrada2,resultado] : noexit : =
        entrada1;entrada2;resultado;stop
      endproc
```

```
(16)  process BFDO_FUNCAO_BINARIA2[entrada1,entrada2,resultado] : noexit : =
        entrada1;entrada2;resultado;p;
        where
            process p ...endproc
      endproc
```

```
(17)  process BFDO_FUNCAO_BINARIA3[entrada1,entrada2,resultado] : noexit : =
        entrada1;entrada2;resultado; BFDO_INTERRUPCAO1[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : =
                interrompe;motivo;stop
      endproc
```

```
(18)  process BFDO_FUNCAO_BINARIA4[entrada1,entrada2,resultado] : noexit : =
        entrada1;entrada2;resultado; BFDO_INTERRUPCAO2[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : =
                interrompe;motivo;p
                where
                    process p ... endproc
      endproc
```

```
(19)  process BFDO_INVERTE1[a,b,c] : exit : =
        c;b;a;exit
      endproc
```

```
(20)  process BFDO_INVERTE2[a,b,c] : exit : =
        c;b;a;p;
        where
            process p ... endproc
      endproc
```

```
(21)  process BFDO_INVERTE3[a,b,c] : exit : =
        c;b;a; BFDO_INTERRUPCAO1[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : =
                interrompe;motivo;stop
      endproc
```

```
(22)  process BFDO_ENTRADA[entrada,saida] : noexit : =
        entrada;saida;stop
      endproc
```

```
(23)  process BFDO_INVERTE4[a,b,c] : exit : =
        c;b;a; BFDO_INTERRUPCAO2[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : =
                interrompe;motivo;p
                where
                    process p ... endproc
      endproc
```

```
(24)  specification BFDO_BASICO1[a] : noexit : =
      behaviour
        BFDO_BASICO1[a]
        where
            process BFDO_BASICO1[a] ... endproc
      endspec
```

```
(25)  specification BFDO_BASICO2[a,b] : noexit : =
      behaviour
        BFDO_BASICO2[a,b]
        where
            process BFDO_BASICO2[a,b] ... endproc
      endspec
```

```
(26)  specification BFDO_BASICO3[a,b,c] : noexit : =
      behaviour
        BFDO_BASICO3[a,b,c]
        where
            process BFDO_BASICO3[a,b,c] ... endproc
      endspec
```

```
(27)  specification BFDO_BASICO4[a,b,c,d] : noexit : =
      behaviour
        BFDO_BASICO4[a,b,c,d]
        where
            process BFDO_BASICO4[a,b,c,d] ... endproc
      endspec
```

```
(28)  specification BFDO_BASICO5[a,b,c,d,e] : noexit : =
      behaviour
        BFDO_BASICO5[a,b,c,d,e]
        where
            process BFDO_BASICO5[a,b,c,d,e] ... endproc
      endspec
```

3.3.1.2 Construções Predefinidas em LOTOS Básico, processos Finitos, Indeterminísticos e com somente eventos Observáveis (BFIO)

Nesta seção são apresentadas trinta e uma construções predefinidas em LOTOS Básico, cujos processos são finitos, indeterminísticos e com somente eventos observáveis. Estas construções predefinidas são classificadas como BFIO.

```
(29)  process BFIO_ESCOLHA1[a1,a2] : noexit :=
        a1; stop [ ] a2; stop
      endproc
```

```
(30)  process BFIO_ESCOLHA2[a1,a2] : noexit :=
        a1; BFDO_INTERRUPCAO1[interrompe,motivo] [ ] a2; stop
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(31)  process BFIO_ESCOLHA3[a1,a2] : noexit :=
        a1; stop [ ] a2; BFDO_INTERRUPCAO1[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(32)  process BFIO_ESCOLHA4[a1,a2] : noexit :=
              a1;BFDO_INTERRUPCAO1[interrompe,motivo]
        [ ]   a2;BFDO_INTERRUPCAO1[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(33)  process BFIO_ESCOLHA5[a1,a2] : noexit :=
        a1; BFDO_INTERRUPCAO2[interrompe,motivo] [ ] a2; stop
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(34)  process BFIO_ESCOLHA6[a1,a2] : noexit :=
              a1;BFDO_INTERRUPCAO2[interrompe,motivo]
          [ ] a2;BFDO_INTERRUPCAO2[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(35)  process BFIO_ESCOLHA7[a1,a2] : noexit :=
        a1; stop [ ] a2; BFDO_INTERRUPCAO2[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(36)  process BFIO_ESCOLHA8[a1,a2] : noexit :=
        a1; a3; stop [ ] a2; a3;stop
      endproc
```

```
(37)  process BFIO_HABILITA1[a1,a2] : noexit :=
        a1; p [ ] a2; p
        where
            process p... : noexit : =...endproc
      endproc
```

```
(38)  process BFIO_HABILITA2 [a1,a2] : exit : =
        a1; exit [ ] a2; exit
      endproc
```

```
(39)  process BFIO_HABILITA3 [a1,a2] : exit : =
        a1; BFDO_INTERRUPCAO1[interrompe,motivo] [ ] a2; exit
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(40)  process BFIO_HABILITA4[a1,a2] : exit : =
        a1; exit [ ] a2; BFDO_INTERRUPCAO1[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(41)  process BFIO_HABILITA5[a1,a2] : exit : =
              a1; BFDO_INTERRUPCAO1[interrompe,motivo]
          [ ] a2; BFDO_INTERRUPCAO1[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(42)  process BFIO_HABILITA6 [a1,a2] : exit : =
        a1; BFDO_INTERRUPCAO2[interrompe,motivo] [ ] a2; exit
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(43)  process BFIO_HABILITA7[a1,a2] : exit : =
        a1; exit [ ] a2; BFDO_INTERRUPCAO2[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(44)  process BFIO_HABILITA8[a1,a2] : exit : =
            a1;BFDO_INTERRUPCAO2[interrompe,motivo]
          [ ] a2;BFDO_INTERRUPCAO2[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : = ... endproc
      endproc
```

```
(45)  process BFIO_HABILITA_DESABILITA1[a1, a2] : noexit : =
        a1; exit [ ] a2;stop
      endproc
```

```
(46)  process BFIO_HABILITA_DESABILITA2[a1, a2] : noexit : =
        a1; BFDO_INTERRUPCAO1[interrompe,motivo] [ ] a2;stop
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : = ... endproc
        endproc
```

```
(47)  process BFIO_HABILITA_DESABILITA3[a1, a2] : noexit : =
        a1; exit [ ] a2; BFDO_INTERRUPCAO1[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO1[interrompe,motivo] : noexit : = ... endproc
        endproc
```

```
(48)  process BFIO_HABILITA_DESABILITA4[a1, a2] : noexit : =
        a1; BFDO_INTERRUPCAO2[interrompe,motivo] [ ] a2;stop
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : = ... endproc
        endproc
```

```
(49)  process BFIO_HABILITA_DESABILITA5[a1, a2] : noexit : =
        a1; exit [ ] a2; BFDO_INTERRUPCAO2[interrompe,motivo]
        where
            process BFDO_INTERRUPCAO2[interrompe,motivo] : noexit : = ... endproc
        endproc
```

```
(50)  process BFIO_TRI_SEPARADOR[entrada,saida1,saida2,saida3] : noexit : =
            entrada; (saida1;stop
                    [ ] saida2;stop
                    [ ] saida3;stop)
      endproc
```

```
(51)  process BFIO_TRI_SEPARADOR[entrada,saida1,saida2,saida3] : noexit : =
          entrada; (saida1; BFDO_INTERRUPCAO1[interrompe,motivo]
                   [ ] saida2; BFDO_INTERRUPCAO1[interrompe,motivo]
                   [ ] saida3; BFDO_INTERRUPCAO1[interrompe,motivo])
      endproc
```

```
(52)  process BFIO_TRI_SEPARADOR[entrada,saida1,saida2,saida3] : noexit : =
          entrada; (saida1; BFDO_INTERRUPCAO2[interrompe,motivo]
                   [ ] saida2; BFDO_INTERRUPCAO2[interrompe,motivo]
                   [ ] saida3; BFDO_INTERRUPCAO2[interrompe,motivo])
      endproc
```

```
(53)  process BFIO_COMUTLIGADO[liga,entrada,estimulo1,saida1,estimulo2,saida2] :noexit: =
        liga;entrada; (estimulo1;saida1;stop
                   [ ] estimulo2,saida2;stop)
      endproc
```

```
(54)  process BFIO_COMUTLIGADO1[liga,entrada,estimulo1,saida1,estimulo2,saida2]:noexit: =
        liga;entrada; (estimulo1;saida1; BFDO_INTERRUPCAO1[interrompe,motivo]
                   [ ] estimulo2,saida2;stop)
      endproc
```

```
(55)  process BFIO_COMUTLIGADO2[liga;entrada,estimulo1,saida1,estimulo2,saida2]:noexit: =
        liga;entrada; (estimulo1;saida1; stop
                   [ ] estimulo2,saida2; BFDO_INTERRUPCAO1[interrompe,motivo])
      endproc
```

```
(56)  process BFIO_COMUTLIGADO3[liga,entrada,estimulo1,saida1,estimulo2,saida2]:noexit: =
        liga;entrada; (estimulo1;saida1; BFDO_INTERRUPCAO2[interrompe,motivo]
                   [ ] estimulo2,saida2;stop)
      endproc
```

```
(57)  process BFIO_COMUTLIGADO4[liga,entrada,estimulo1,saida1,estimulo2,saida2]:noexit: =
        liga;entrada; (estimulo1;saida1; stop
                   [ ] estimulo2,saida2; BFDO_INTERRUPCAO2[interrompe,motivo])
      endproc
```

```
(58)  process BFIO_COMUTLIGADO5[liga,entrada,estimulo1,saida1,estimulo2,saida2]:noexit: =
        liga;entrada; (estimulo1;saida1; BFDO_INTERRUPCAO2[interrompe,motivo]
                   [ ] estimulo2,saida2; BFDO_INTERRUPCAO2[interrompe,motivo])
      endproc
```

```
(59)  process BFIO_PARALELO[entrada1,entrada2,saida1,saida2]:noexit: =
        p[entrada1,saida1] ||| p[entrada2,saida2]
      where
        process p ... endproc
      endproc
```

3.3.1.3 Construções Predefinidas em LOTOS Básico, processos Infinitos, Determinísticos e com somente eventos Observáveis (BIDO)

Nesta seção são apresentadas onze construções predefinidas em LOTOS Básico, cujos processos são infinitos, determinísticos e com somente eventos observáveis. Estas construções predefinidas são classificadas como BIDO.

```
(60)  process BIDO_CICLICO1[a] : noexit : =
        a;BIDO_CICLICO1[a]
      endproc
```

```
(61)  process BIDO_CICLICO2[a1,a2] : noexit : =
        a1;a2;BIDO_CICLICO2[a1,a2]
      endproc
```

```
(62)  process BIDO_CICLICO3[a1,a2,a3] : noexit : =
        a1;a2;a3;BIDO_CICLICO3[a1,a2,a3]
      endproc
```

```
(63)  process BIDO_CICLICO4[a1,a2,a3,a4] : noexit : =
        a1;a2;a3;a4;BIDO_CICLICO[a1,a2,a3,a4]
      endproc
```

```
(64)  process BIDO_FUNCAO[dado,resultado] : noexit :=
        dado;resultado; BIDO_FUNCAO[dado,resultado]
      endproc
```

```
(65)  process BIDO_CONECTOR[a,b,c] : noexit :=
        a;b;c;BIDO_CONECTOR[a,b,c]
      endproc
```

```
(66)  process BIDO_ESR [entrada,saida,reabre] : noexit : =
        entrada;saida;reabre; BIDO_ESR[entrada,saida,reabre]
      endproc
```

```
(67)  process BIDO_DISPARADOR[entrada,estimulo,saida] : noexit : =
        entrada;estimulo;saida; BIDO_DISPARADOR[entrada,estimulo,saida]
      endproc
```

```
(68)  process BIDO_FONTE[liga,sai-valor] : noexit : =
        liga;sai-valor; BIDO_FONTE[liga,sai-valor]
      endproc
```

```
(69)  process BIDO_SUMIDOURO[entrada] : noexit : =
        entrada; BIDO_SUMIDOURO[entrada]
      endproc
```

```
(70)  process BIDO_CALCULATE[collected_data,calculated_data] : noexit : =
        collected_data;calculated_data;calculated_data;collected_data;
        CALCULATE[collected_data,calculated_data]
      endproc
```

```
(71)  specification BIDO_PROV_COM[a,b] : noexit
      behavior
        BIDO_PROV_COM[a,b]
      where
        process BIDO_PROV_COM[a,b] : noexit :=
              a;b;BIDO_PROV_COM[a,b]
        endproc
      endspec
```

3.3.1.4 Construções Predefinidas em LOTOS Básico, processos Infinitos, Indeterminísticos e com somente eventos Observáveis (BIIO)

Nesta seção são apresentadas vinte construções predefinidas em LOTOS Básico, cujos processos são infinitos, indeterminísticos e com somente eventos observáveis. Estas construções predefinidas são classificadas como BIIO.

```
(72)  process BIIO_DISJUNTOR[a1,a2,a3] : noexit : =
        a1; (a2;BIIO_DISJUNTOR[a1,a2,a3]
      [ ] a3; BIIO_DISJUNTOR[a1,a2,a3] )
      endproc
```

```
(73)  process BIIO_OPCAO[a,b] : noexit : =
        a;exit [ ] ; BIIO_OPCAO[a,b]
      endproc
```

```
(74)  process BIIO_COMUTADOR[entrada,estimulo1,saida1,estimulo2,saida2] : noexit : =
        entrada;(estimulo1;saida1;
                    BIIO_COMUTADOR[entrada,estimulo1,saida1,estimulo2,saida2]
                 [ ] estimulo2;saida2;
                    BIIO_COMUTADOR[entrada,estimulo1,saida1,estimulo2,saida2])
      endproc
```

```
(75)  process BIIO_SEPARADOR[entrada,saida1,saida2] : noexit : =
        entrada;(saida1; BIIO_SEPARADOR[entrada,saida1,saida2] [ ] saida2;stop)
      endproc
```

```
(76)  process BIIO_SEPARADOR-LIGADO[liga,entrada,saida1,saida2] : noexit : =
        liga;entrada;(saida1;BIIO_SEPARADOR-LIGADO[liga,entrada,saida1,saida2]
                    [ ] saida2;stop)
      endproc
```

```
(77)  process BIIO_SEPARADOR_REABERTO[entrada,saida1,reabre,saida2] : noexit : =
        entrada;(saida1;reabre;
                    BIIO_SEPARADOR_REABERTO[entrada,saida1,reabre,saida2]
                 [ ] saida2,stop)
      endproc
```

```
(78)  process BIIO_INTERATIVO[a,b,c] : noexit : =
        a; BIIO_INTERATIVO[a,b,c];b;stop [ ] c;stop
      endproc
```

```
(79)  process BIIO_LOC_M[mo_operation,mo_notification] : noexit : =
        mo_operation; BIIO_LOC_M[mo_operation,mo_notification]
        [ ]
        mo_notification; BIIO_LOC_M[mo_operation,mo_notification]
      endproc
```

```
(80)  process BIIO_COMPARE[notif,baseline_data,calculated_data] : noexit : =
        calculated_data;baseline_data;
          (i;calculated_data;BIIO_COMPARE[notif,baseline_data,calculated_data]
        [ ] i;notif;calculated_data; BIIO_COMPARE[notif,baseline_data,calculated_data])
      endproc
```

```
(81)  process BIIO_PARALELO[a,b,c,d] : noexit : =
        p[a,b,c,d] || p1[a,b,c,d]
      where
        process p  ...endproc
        process p1 ...endproc
      endproc
```

```
(82)  process BIIO_COMPOSIÇÃO[entrada,m1,m2,saída] : noexit : =
        p[entrada,m1] | [m1] | p1[m1,m2] | [m2] | p2[m2,saida]
      where
        process p  ...endproc
        process p1 ...endproc
        process p2 ...endproc
      endproc
```

```
(83)  process BIIO_SEQ_PROC[a,b,c,d] : noexit : =
        p[a,b] >> p1[c,d]
      where
        process p  ...endproc
        process p1 ...endproc
      endproc
```

```
(84)  process BIIO_SEQ_PROC[a,b,c,d,e] : noexit : =
        p[a,b,c] [> p1[d,e]
      where
        process p  ...endproc
        process p1 ...endproc
      endproc
```

```
(85)  specification BIIO_Multiway1[b,c] : noexit
      ...
      behavior
        Proc1[a] |[a]| Proc2[a]
      ...
      endspec
```

```
(86)  specification BIIO_Multiway2[b,c] : noexit
      ...
      behavior
        a; Proc1[a] |[a]| a;Proc2[a] |[a] | a;Proc3[a]
      ...
      endspec
```

```
(87)  specification BIIO_HIDE1[a,b] : noexit
      ...
      behavior
        hide c in Proc1[b,c] |[c]| Proc2[b,c,a]
      ...
      endspec
```

```
(88)  specification BIIO_HIDE2[a,c] : noexit
      ...
      behavior
        hide a1;a2;b in Proc1[a;a1;a2] |[a1;a2]| Proc2[a1;b] |[b]| Proc3[a2;b;c]
      ...
      endspec
```

```
(89)  specification BIIO_AgenteP[msg,op_ger,not_ag,op_obj,not_obj] : noexit
      ...
      behavior
        BIIO_COORD[msg] |[msg]| BIIO_PRO_COM[msg,op_ger,not_ag,op_obj,not_obj]
      where
      process BIIO_COORD[msg] : noexit : = .... endproc
      process BIIO_PRO_COM[msg,op_ger,not_ag,op_obj,not_obj] : noexit : = .... endproc
      ...
      endspec
```

```
(90)  process BIIO_COORD[msg] : noexit : =
      msg;BIIO_COORD[msg]
      [ ]
      msg;BIIO_PRO_COM[msg,op_ger,not_ag,op_obj,not_obj]
      endproc
```

```
(91)  process BIIO_PRO_COM[msg,op_ger,not_ag,op_obj,not_obj] : noexit : =
      msg;BIIO_COORD[msg]
      [ ]
      op_ger;op_obj;BIIO_PRO_COM[msg,op_ger,not_ag,op_obj,not_obj]
      [ ]
      not_obj;not_ag;BIIO_PRO_COM[msg,op_ger,not_ag,op_obj,not_obj]
      endproc
```

3.3.1.5 Construções Predefinidas em LOTOS Básico, processos Infinitos, Indeterminísticos e com eventos Internos (BIII)

Nesta seção são apresentadas doze construções predefinidas em LOTOS Básico, cujos processos são infinitos, indeterminísticos e com eventos internos. Estas construções predefinidas são classificadas como BIII.

```
(92)  process BIII_OPCAO_INTERNA[a,b] : noexit : =
        a; BIII_OPCAO_INTERNA[a,b]  [ ]  i;stop
      endproc
```

```
(93)  process BIII_PREDICADO[entrada,sai-se-true,sai-se-false] : noexit : =
        entrada; (i;sai-se-true; BIII_PREDICADO[entrada,sai-se-true,sai-se-false]
                 [ ] i;sai-se-false; BIII_PREDICADO[entrada,sai-se-true,sai-se-false])
      endproc
```

```
(94)  process BIII_PREDEFINIDO[notif,baseline_data,mo_notif,operat]: noexit :=
                baseline_data;
                 (i; p [] i;notif;p)
      where
        process p  ...endproc
      endproc
```

```
(95)  process BIII_REMOTE_MONITOR[operat,mo_notif,collected_data] : noexit : =
      operat;collected_data;collected_data;
        BIII_REMOTE_MONITOR [operat,mo_notif,collected_data]
      []mo_notif;collected_data;collected_data;
        BIII_REMOTE_MONITOR [operat,mo_notif,collected_data]
      endproc
```

```
(96)  specification BIII_AgenteP[op_ger,not_ag,op_obj,not_obj] : noexit
      ...
      behavior
        op_ger;(op_obj;not_obj:not_ag;BIII_Agente[op_ger,not_ag,op_obj,not_obj]
      [] op_obj;not_obj;(i;not_ag; BIII_Agente[op_ger,not_ag,op_obj,not_obj]
      [] i;BIII_Agente[op_ger,not_ag,op_obj,not_obj]
      ...
      endspec
```

```
(97)  specification BIII_Multiway1[b,c] : noexit
      ...
      behavior
        hide a in Proc1[a] |[a]| Proc2[a] |[a] | Proc3[a]
      ...
      endspec
```

```
(98)  specification BIII_Multiway2[a] : noexit
      ...
      behavior
        a;Proc1[a] |[a]| hide B in a;b;Proc2[b] |[b] | b;Proc3[b]
      ...
      endspec
```

```
(99)  specification BIII_HIDE1[a,b] : noexit
      ...
      behavior
        hide c in Proc1[b,c] |[c]| Proc2[b,c,a]
      ...
      endspec
```

```
(100) specification BIII_HIDE2[a,c] : noexit
      ...
      behavior
        hide a1;a2;b in Proc1[a;a1;a2] |[a1;a2]| Proc2[a1;b] |[b]| Proc3[a2;b;c]
      ...
      endspec
```

```
(101) specification BIII_PROV_COM[a,b,c] : noexit
      behavior
      hide c in
        BIII_ENTIDADE_A[a,c] | [c] | BIII_ENTIDADE_B[c,b]
      where
        process BIII_ENTIDADE_A[a,c]: noexit : =
              a;c;c; BIII_ENTIDADE_A[a,c]
        endproc
        process BIII_ENTIDADE_B[c,b]: noexit : =
              c;b;c; BIII_ENTIDADE_B[c,b]
        endproc
      endspec
```

```
(102) specification BIII_PROV_COM1[a,b,c] : noexit
      behavior
      hide m,n in
        (BIII_ENTIDADE_A[a,m] | | | BIII_ENTIDADE_B[n,b])
        | [m,n] |
        BIII_SERV_SUBJ[m,n]
      where
        process BIII_ENTIDADE_A[a,m]: noexit : =
              a;m;m; BIII_ENTIDADE_A[a,m]
        endproc
        process BIII_ENTIDADE_B[n,b]: noexit : =
              n;b;n; BIII_ENTIDADE_B[n,b]
        endproc
        process BIII_SERV_SUBJ[m,n]: noexit : =
              m;n;n;m; BIII_SERV_SUBJ[m,n]
        endproc
      endspec
```

```
(103)  process BIIO_OCULTAÇÃO[entrada,saída] : noexit : =
           hide m1,m2 in
           entrada [entrada,m1] | [m1] | p1[m1,m2] | [m2] | p2[m2,saida]
       where
           process p  ...endproc
           process p1 ...endproc
           process p2 ...endproc
       endproc
```

3.3.2 Construções Predefinidas em ACT ONE

Estas construções permitem a representação rigorosa de tipos abstratos de dados.

3.3.2.1 Construções Predefinidas em ACT ONE da Biblioteca Padrão (ABP)

Nesta seção são apresentadas vinte e sete construções predefinidas em ACT ONE, as quais fazem parte do Anexo A Biblioteca Padrão LOTOS [ISO 8807].

```
(104)  Boolean
```

```
(105)  Fboolean
```

```
(106)  Element
```

```
(107)  Set
```

```
(108)  BasicNonEmptyString
```

```
(109)  RicherNonEmptyString
```

```
(110)  NonEmptyString
```

```
(111)  String0
```

```
(112)  String1
```

```
(112)  String
```

(113)	BasicNaturalNumber
(114)	NaturalNumber
(115)	NatRepresentations
(116)	HexNatRepr
(117)	HexString
(118)	HexDigit
(119)	DecNatRepr
(120)	DecString
(121)	DecDigit
(122)	OctNatRepr
(123)	OctString
(124)	OctDigit
(125)	BitNatRepr
(126)	BitString
(127)	BitDigit
(128)	Octet
(129)	OctetString

3.3.2.2 Construções Predefinidas em ACT ONE - Objetos Gerenciados (AOG)

Diversos objetos gerenciados podem ter seus comportamentos e dados especificados em LOTOS e armazenados na biblioteca bibLOTOS. Entretanto, devido à complexidade dos estudos

necessários, ainda não foi possível apresentar, neste trabalho, especificações completas. Nesta seção é apresentado como exemplo, o comportamento do objeto TOP.

```
(130)   specification TOP_Class [failure,parameter]: noexit
        behaviour
              TOP[failure,parameter]
        where
        process TOP[failure,parameter]:noexit:=
                failure?error:errorCondition;
                parameter!miscelaneousError;
                TOP[failure,parameter]
        endproc
        endspec
```

3.3.2.3 Construções Predefinidas em ACT ONE - Outros Tipos de Dados (AOT)

Nesta seção é apresentado um esforço para a definição de construções predefinidas em ACT ONE. Seguem dez construções desse tipo.

```
(131)   type ELEMENT is NAT endtype

        type STACK_TYPE(DATA) is
          formal_sorts DATA
          formal_opns ERRORd: --> DATA
          sorts    STACK_TYPE
          opns     NEW_STACK:    --> STACK_TYPE
                   ERRORs:       -->STACK_TYPE
                   PUSH:  STACK_TYPE x DATA --> STACK_TYPE
                   POP:  STACK_TYPE  -->  STACK_TYPE
                   TOP: STACK_TYPE  -->   DATA
                   EMPTY:  STACK_TYPE --> BOOLEAN
          eqns     EMPTY(NEW_STACK) = TRUE
                   EMPTY(PUSH(s,d))  =  FALSE
                   TOP(NEW_STACK)  =  ERRORd
                   TOP(PUSH(s,d)) = d
                   POP(NEW_STACK)  =  ERRORs
                   POP(PUSH(s,d))  =  s
        endtype
```

(132)
```
type  STACK_TYPE(ELEMENT) is
   STACK_TYPE(DATA)  actualized_by ELEMENT
   unsign sortnames ELEMENT for DATA
      opnames  ERRORd for ERRORd
endtype
```

(133)
```
type Nat_Number_Queue is Nat_Numbers
sorts queue
opns  create: -> queue
   add: Nat, queue -> queue
   first: queue -> Nat
   remove: queue -> queue
eqns forall x, y: Nat, z: queue

ofsort nat
   first(create) = 0;
   first(add(x,create)) = x;
   first(add(x,add(y,z))) = first(add(y,z));

ofsort queue
   remove(create) = create;
remove(add(x,create)) = create;
remove(add(x,add(y,z))) = add(x,remove(add(y,z)));
endtype
```

(134)
```
type Character_Queue is Character
sorts queue
opns   create: -> queue
   add: -> char, queue -> queue
   first: -> queue -> queue
remove: -> queue
eqns forall x, y: Char, z: queue

ofsort Char
   first(create) = e;
   first(add(x,create)) = x;
   first(add(x,add(y,z))) = first(add(y,z));

ofsort queue
   remove(create) = create;
   remove(add(x,create)) = create;
   remove(add(x,add(y,z))) = add(x,remove(add(y,z)));
endtype
```

(135)
```
type Queue is
formalsorts data
formalopns d0: -> data
sorts queue
opns   create: -> queue
    add: data, queue -> queue
    first: queue -> queue
    remove: queue -> queue
eqns forall x, y: data, z: queue

ofsort data
    first(create) = d0;
    first(add(x,create)) = x;
    first(add(x,add(y,z))) = first(add(y,z));

ofsort queue
    remove(create) = create;
    remove(add(x,create)) = create;
    remove(add(x,add(y,z))) = add(x,remove(add(y,z)));
endtype
```

(136)
```
type Nat_Numbers_Queue is
    Queue actualizedby Nat_Numbers using
    sortnames nat for data
    opnnames 0 for d0
endtype

type Character_Queue is
    Queue actualizedby Characters using
    sortnames Char for Data
    opnnames e for d0
endtype
```

(137)
```
type Extra_Queue is
formalsort data
formalopns    d0: -> data
        _*_ : data, data -> data
formaleqns forall x,y: data

ofsort queue
opns create: -> queue

endtype
```

```
(138)   type Connection is
          Queue renamedby
          sortnames    channel for queue
                       objects for data
          opnnames send for add
                   receive for first
        endtype
```

3.3.3 Construções Predefinidas em LOTOS Completo

Nesta seção são apresentadas duas construções predefinidas em LOTOS Completo.

```
(139)   specification Gerente_Agente[sucesso, falha]:noexit
          library BOOLEAN endlib
          type operations is BOOLEAN
            (* sorts op
            opns get,set:->op*)
          endtype
          behaviour
          hide comando1, comando2 in
          Gerente[comando1]
          |[comando1]|
          Agente[comando1, comando2]
          |[comando2]|
          MO[comando2]
        where
          process Gerente[comando]:noexit:=
          hide user in
            user?x:bool; comando!x ; Gerente[comando]
          endproc

          process Agente[comando1, comando2]:noexit:=
            comando1?x:bool;comando2!x;
            Agente[comando1, comando2]
          endproc

          process MO[comando]:noexit:=
            comando?x:bool; (    [x=TRUE] -> i; MO[comando]
                              [] [x=FALSE] -> i; MO[comando])
          endproc
        endspec
```

```
(140)   process STACK[push,pop,top,empty] :=
        empty!EMPTY(NEW_STACK);
        USED_STACK[push,pop,top,empty](NEW_STACK)
        [] push ?x: ELEMENT;
        USED_STACK[push,pop,top,empty](PUSH(NEW_STACK,x))
        where
            process USED_STACK[push,pop,top,empty] (s:STACK_TYPE)
           push ? x: ELEMENT; USED_STACK[push,pop,top,empty](PUSH(s,x))
                [] [NOT(EMPTY(s))] --> pop ! SIGNAL;
                  USED_STACK[push,pop,top,empty] (POP(s))
                [] [NOT(EMPTY(s))] --> top ! TOP(s);
                  USED_STACK[push,pop,top,empty] (s)
              [] empty ! EMPTY(s); USED_STACK[push,pop,top,empty] (s)
           endproc
        endproc
        endspec
```

4. A Biblioteca bibLOTOS

"Broad-spectrum environments integrate tightly coupled methods and tools that, when properly applied, establish a complete environment for building or supporting a software application" [Evan 89].

Neste capítulo apresenta-se a biblioteca bibLOTOS. Esta biblioteca armazena um conjunto de construções predefinidas LOTOS, disponíveis para reutilização, as quais especificam comportamentos, estruturas de dados, ou ambos.

Na bibLOTOS tais construções (neste trabalho representadas por especificações LOTOS) vêm acompanhadas de informações adicionais (como, por exemplo, descrições informais em linguagem natural) que podem ser úteis nas tarefas de organização, localização, utilização das construções predefinidas, enfim, auxiliar nas tarefas do projetista.

Como a bibLOTOS armazena informações variadas, faz-se necessária a organização destas informações. Para isso, é conveniente a utilização de um programa específico, ou seja, um Gerenciador de Banco de Dados. O banco de dados no qual a bibLOTOS foi implementada é o *Microsoft Access©*.

O esforço para a concepção de construções predefinidas orienta-se para a produção de construções de uso geral. Ao adotar a abordagem que faz uso de construções predefinidas, os usuários da bibLOTOS devem sentir-se encorajados a ampliar o seu acervo, sendo que diversas especificações LOTOS podem ser obtidas a partir das especificações já armazenadas na biblioteca. Sugestões de encorajamento nesse sentido, para outra biblioteca de construções predefinidas são também apresentadas em [BoVa 95].

Nas próximas seções deste capítulo apresenta-se o projeto de concepção da biblioteca bibLOTOS. Em seguida são apresentados os detalhes de implementação desta. Por fim faz-se uma apresentação das principais telas da biblioteca.

4.1 Concepção da bibLOTOS

Para cada construção predefinida, especificada em LOTOS, e presente na bibLOTOS, tem-se armazenado um conjunto de informações associadas. O propósito de tais informações é oferecer ao projetista uma ferramenta que possibilite simplificar o processo de elaboração de uma especificação LOTOS. Essa ferramenta deve ser de fácil utilização e deve integrar representações gráficas e textuais. Veja a Figura 4.1.

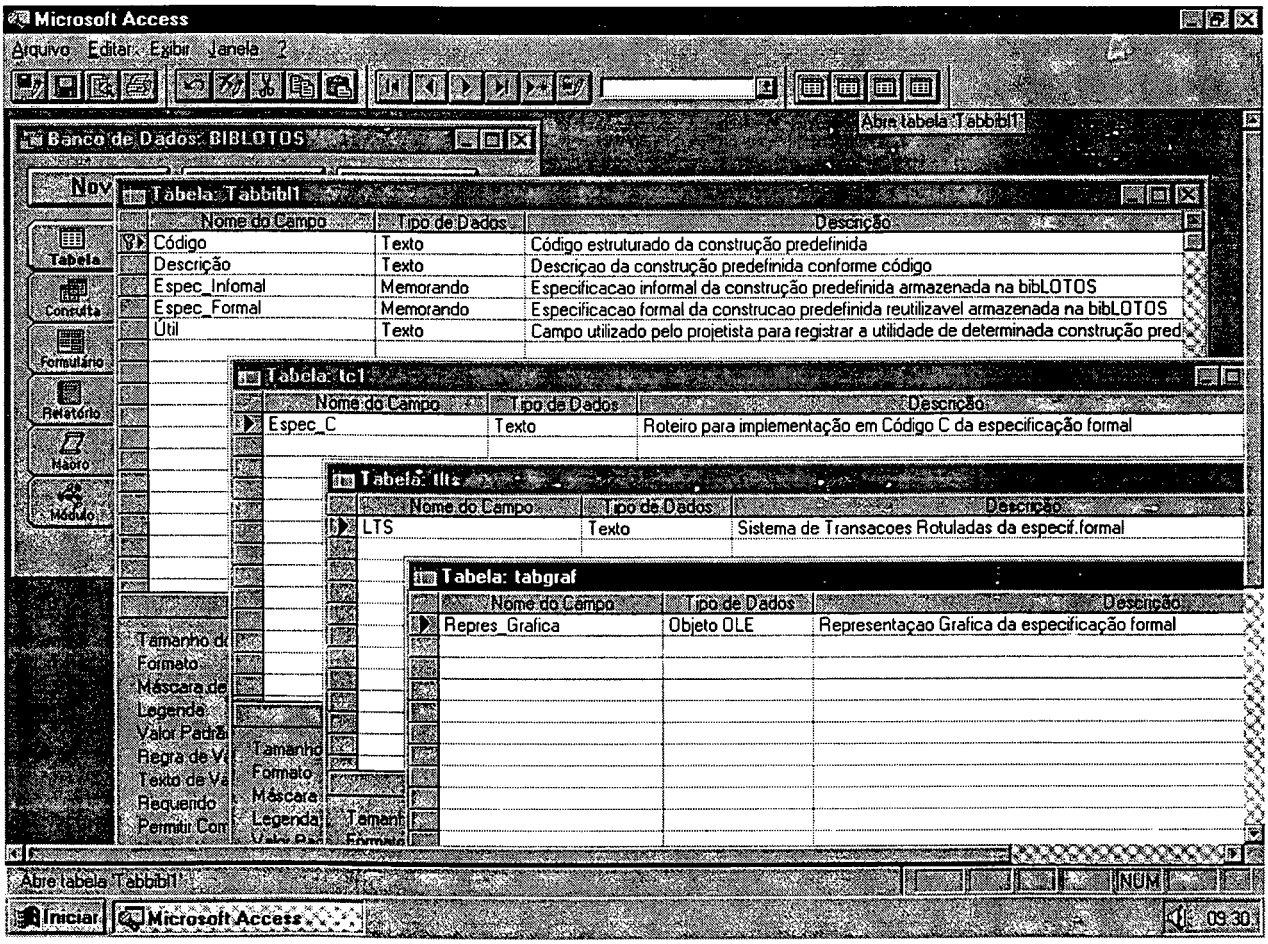


Figura 4.1 - Gerenciador de Banco de Dados *Microsoft Access*© e tabelas da bibLOTOS.

Conforme apresentado na Figura 4.1, a bibLOTOS é implementada com os seguintes campos:

⇨ **Código**: campo de dado tipo texto, o qual armazena o código estruturado da construção predefinida. Após identificar a construção predefinida conforme o tipo com quatro *caracteres* específicos, o projetista insere um "_" e anexa um nome ao processo.

Este campo é a chave primária da bibLOTOS. Ela possibilita identificar de um modo único cada registro individual armazenado no banco de dados. No *Microsoft Access* não é permitido o uso de duplicatas ou valores nulos em um campo de chave primária.

⇨ **Descrição**: campo de dado tipo texto, que identifica a construção predefinida conforme o código.

⇨ **Espec_informal**: campo de dado tipo texto, que descreve informalmente a construção predefinida.

⇨ **Espec_formal**: campo de dado tipo texto, que armazena a construção predefinida em LOTOS, já validada com o uso de ferramentas. As informações contidas neste campo serão recolhidas para reutilização.

⇨ **Espec_c**: campo de dado tipo texto, no qual é fornecido um roteiro para implementação em código C da especificação correspondente em LOTOS. Este código C é gerado automaticamente com o auxílio da ferramenta TOPO e transferido para a bibLOTOS.

⇨ **LTS**: campo dado tipo texto, onde um sistema de transições rotuladas correspondente à especificação é apresentado (válido apenas para LOTOS Básico e LOTOS completo).

⇨ **Repres_gráfica**: campo tipo objeto OLE (*Object Linking and Embeding)*, no qual representa-se graficamente a construção predefinida. Para esta representação usa-se a linguagem G-LOTOS, que permite a representação gráfica de especificações LOTOS. A representação gráfica facilita a escolha e a seleção de construções predefinidas na fase de descrição e análise de um sistema.

⇨ **Util**: campo tipo texto, que contém um histórico da utilização da construção predefinida em um sistema e que o projetista vai alimentando durante o uso da biblioteca.

A definição do escopo das telas do sistema, bem como da navegação entre elas, foi realizada durante a implementação da biblioteca.

4.2 Implementação da bibLOTOS

A bibLOTOS está implementada no ambiente **LOWE** (*LOTOS Windows Environment*). Fazem parte deste ambiente para microcomputadores as seguintes ferramentas: 1) ***edLOTOS***: um editor de especificações LOTOS; 2) ***bibLOTOS***: uma biblioteca de construções predefinidas LOTOS; e 3) ***TranSP***: um transformador automático de especificações formais de serviço em especificações formais de protocolo. Outras ferramentas para o ambiente integrado LOWE serão futuramente desenvolvidas, a saber: 1) ***Sschecker***: um analisador sintático e semântico; 2) ***TSVvalidator***: um validador para teste, simulação e verificação; e 3) ***Cgenerator***: um tradutor para código C. Veja a Figura 4.2.

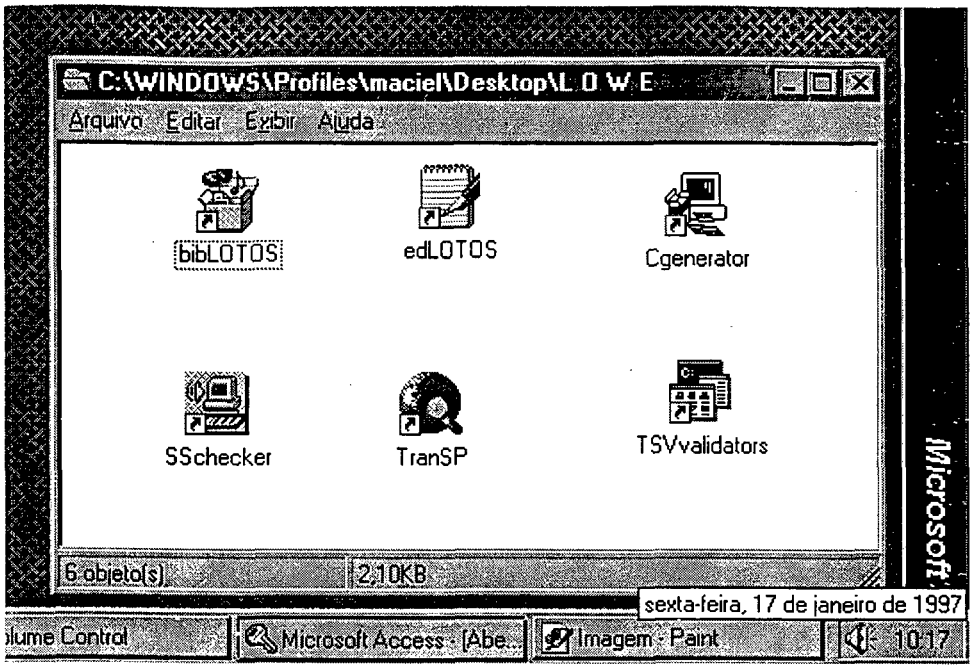


Figura 4.2.- Ambiente de ferramentas LOWE onde está integrada a bibLOTOS.

A biblioteca bibLOTOS armazena construções predefinidas que auxiliam no desenvolvimento de especificações. Para a implementação dessa biblioteca é utilizado o Sistema Gerenciador de Banco de Dados (SGBD) relacional para *Windows Microsoft Access*©.

O *Access* é um poderoso Sistema de Gerenciamento de Banco de Dados relacional para Windows. Ele é totalmente interativo, desde a criação das tabelas de dados até a elaboração de formulários de entrada de dados e relatórios. O *Access* possui ferramentas que permitem facilidades de compartilhamento de informações com outros programas, além de propiciar facilidades na organização e armazenamento de dados que se relacionam entre si. O *Access* inaugurou um novo conceito ao utilizar elementos visuais para a comunicação com o usuário [Alve 93].

Os gerenciadores de banco de dados existentes até então, para ambiente DOS, sempre se valeram de comandos, que deveriam ser memorizados e digitados pelo usuário, quando se desejava

executar alguma tarefa, como por exemplo, criar uma base de dados. No *Access*, qualquer ação que se queira desempenhar sobre uma base de dados, como uma consulta, por exemplo, é realizada através de menus de opções e botões de comandos. Isso se deve ao fato de que o *Access* faz uso dos recursos gráficos do *Windows*.

O *Access* se utiliza de tabelas para armazenar os dados do usuário. Sendo assim, o primeiro passo a ser dado foi a criação de todas as tabelas de dados que fazem parte do banco de dados. Para criação das tabelas foram utilizados os campos determinados na fase de concepção da biblioteca [MaNo 96a]. Uma **tabela** é um conjunto de dados referentes a um assunto, onde cada coluna é denominada Campo e cada linha Registro. Veja a Figura 4.3.

Figura 4.3 - Tabelas da bibLOTOS.

Em seguida criaram-se os relacionamentos entre as tabelas e as consultas de dados. Uma **consulta** pode ser vista como uma pergunta que o usuário faz sobre os dados contidos no banco de dados, que podem estar armazenados em uma ou várias tabelas, tal como a pergunta, "quais construções predefinidas representam dados?". Veja a Figura 4.4.

Figura 4.4 - Consultas da bibLOTOS.

As telas de entrada de dados, denominadas formulários, foram então criadas, e, por último, os relatórios. Esses últimos itens, por outro lado, são o ponto forte do *Access*. Para se criar um formulário ou um relatório pode-se utilizar os Assistentes, que fazem grande parte do trabalho. Um **formulário** é uma disposição das informações na tela para alteração, visualização ou inserção de registros no banco de dados de forma conveniente e com um aspecto mais elaborado. O **relatório**, por sua vez, é a apresentação dos dados na forma impressa. Veja a Figura 4.5.

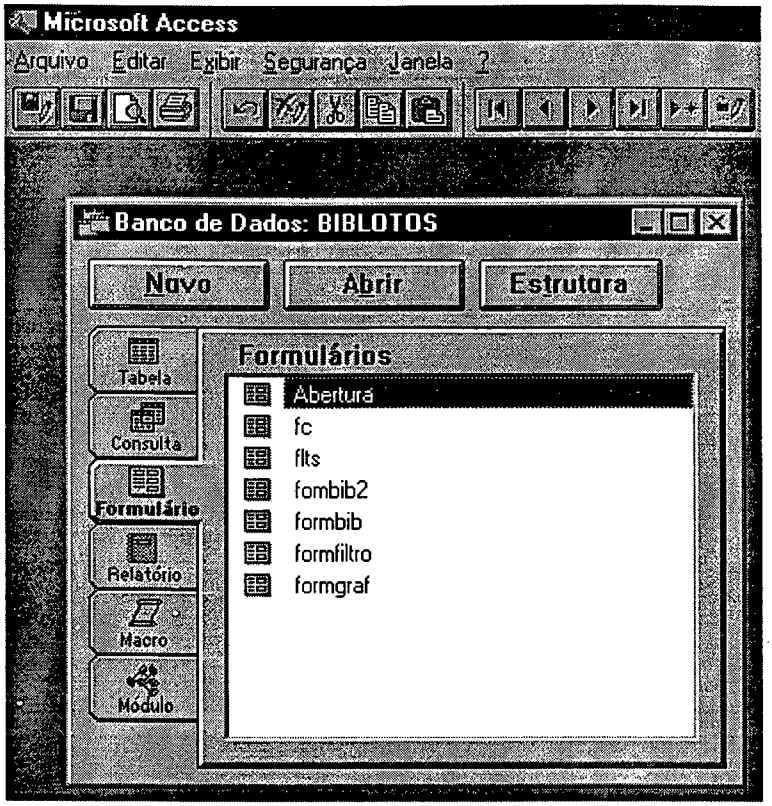


Figura 4.5 - Formulários da bibLOTOS.

O **Consultor** é uma importante característica do *Access*. Enquanto os Assistentes entregam relatórios e formulários padronizados, o Consultor orienta como fazer esse trabalho, além de explicar como criar tabelas, consultas, etc.

O *Access* possui também um recurso bastante poderoso, que possibilita automatizar algumas tarefas. São as **macros**. Pode-se compará-las a pequenas rotinas de programação que podem ser atribuídas a botões. Foram utilizadas diversas macros, como por exemplo, aquelas associadas aos botões de navegação entre as telas da bibLOTOS. O gerenciamento mais especializado dos dados no *Access* é realizado através da linguagem de programação de banco de dados, o *Access Basic*. Veja a Figura 4.6.

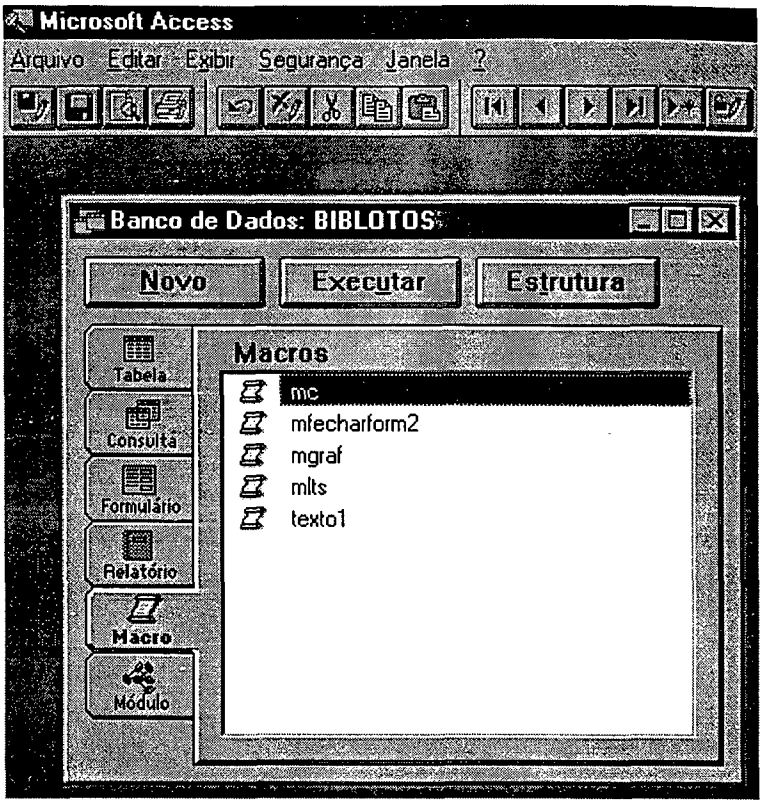


Figura 4.6 - Macros da bibLOTOS.

A estrutura de banco de dados do *Access* combina tabelas contendo dados, consultas, formulários, relatórios, macros e códigos do *Access Basic* em apenas um arquivo `.MDB` (nome da extensão de um arquivo *Access*).

A *Microsoft* possui um *software*, denominado *Microsoft ADT 2.0*, responsável pela geração de código de instalação para um Banco de Dados do *Access* 2.0, que neste caso é executado através de um *run-time*, não sendo necessário acessar o *Access* 2.0 completo para usar um Banco de Dados com a extensão `.MDB`.

A escolha do *Access* para implementação da bibLOTOS justifica-se pelo conjunto dos recursos mencionados acima. Além disso o *Microsoft Access*® é um aplicativo do ambiente *Windows* de uso generalizado.

4.3 Apresentação da bibLOTOS

Nesta seção são exibidas as principais telas da bibLOTOS e um breve roteiro para navegação entre elas. A tela inicial da biblioteca possui três botões automáticos, a saber:

1. **Consultar**: é o botão de chamada a tela de consulta, a qual permite visualizar e recolher as construções predefinidas armazenadas na bibLOTOS.

2. **Cadastrar**: é o botão de chamada para a tela de cadastro, a qual permite a inserção ou a modificação de construções predefinidas na biblioteca.

3. **Sair**: é o botão de saída da biblioteca, retornando ao ambiente *Windows*. Veja a Figura 4.7.

Figura 4.7 - Tela de Abertura da bibLOTOS.

A árvore estruturada, tida como fundo da tela inicial do sistema, exibe a classificação dos códigos estruturados criados para a bibLOTOS. Esta árvore auxilia o projetista no momento em que ele cadastra uma nova construção predefinida, ou até mesmo quando efetua a busca àquelas armazenadas, de modo que ele pode retornar à tela inicial e visualizar a estrutura da biblioteca.

Na **tela de consulta** são apresentados todos os campos definidos na fase de concepção da biblioteca, sendo que o acesso à representação gráfica, ao Sistema de Transações Rotuladas (LTS) e ao código C, referente a cada construção, é realizado através de botões automáticos. Por ser uma tela de consulta não é permitida a edição (modificação) das construções predefinidas exibidas nesta tela. A principal função desta tela é a consulta às construções predefinidas armazenadas na bibLOTOS, com a finalidade de efetuar a reutilização destas no editor de especificações edLOTOS.

Para selecionar um conjunto de construções predefinidas que compartilham das mesmas características, ou para selecionar uma construção em particular, o projetista utiliza o botão automático filtro - 

. Uma caixa de diálogo do *Windows* irá se sobrepor à tela de consulta, solicitando que seja efetuada a entrada do parâmetro para seleção da construção. O projetista, por exemplo, poderia querer uma construção que representasse a estrutura de dados do dígito binário. Conforme o método proposto no Capítulo 5 deste trabalho, o projetista iria utilizar o parâmetro ABP (*ACT ONE* Biblioteca Padrão) para filtrar as construções candidatas a reutilização. Veja a Figura 4.8.

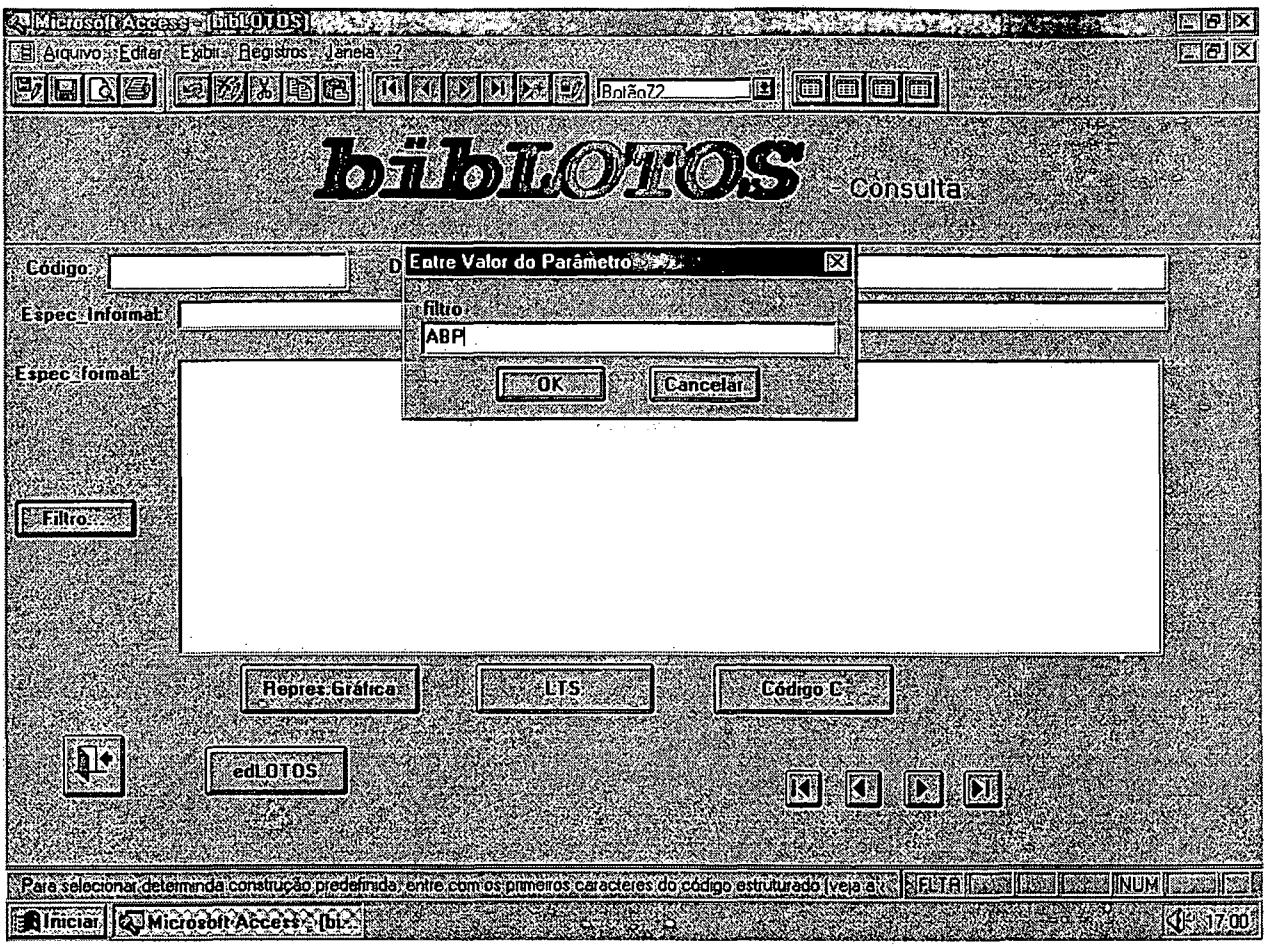


Figura 4.8 - Tela de Consulta/Filtro.

O resultado desta pesquisa é exibido em uma caixa de diálogo do *Windows*, contendo os códigos das construções e as respectivas descrições. O projetista, com o *mouse*, seleciona a construção predefinida de seu interesse. Veja a Figura 4.9.

Figura 4.9 - Tela dos resultados da consulta.

O botão **OK** confirma a consulta à(s) construção(ões) predefinida(s) escolhida(s), apresentando então todos os campos que estão vinculados a ela. Acionando o botão **Cancelar** o projetista volta à tela de consulta. Veja a Figura 4.10.

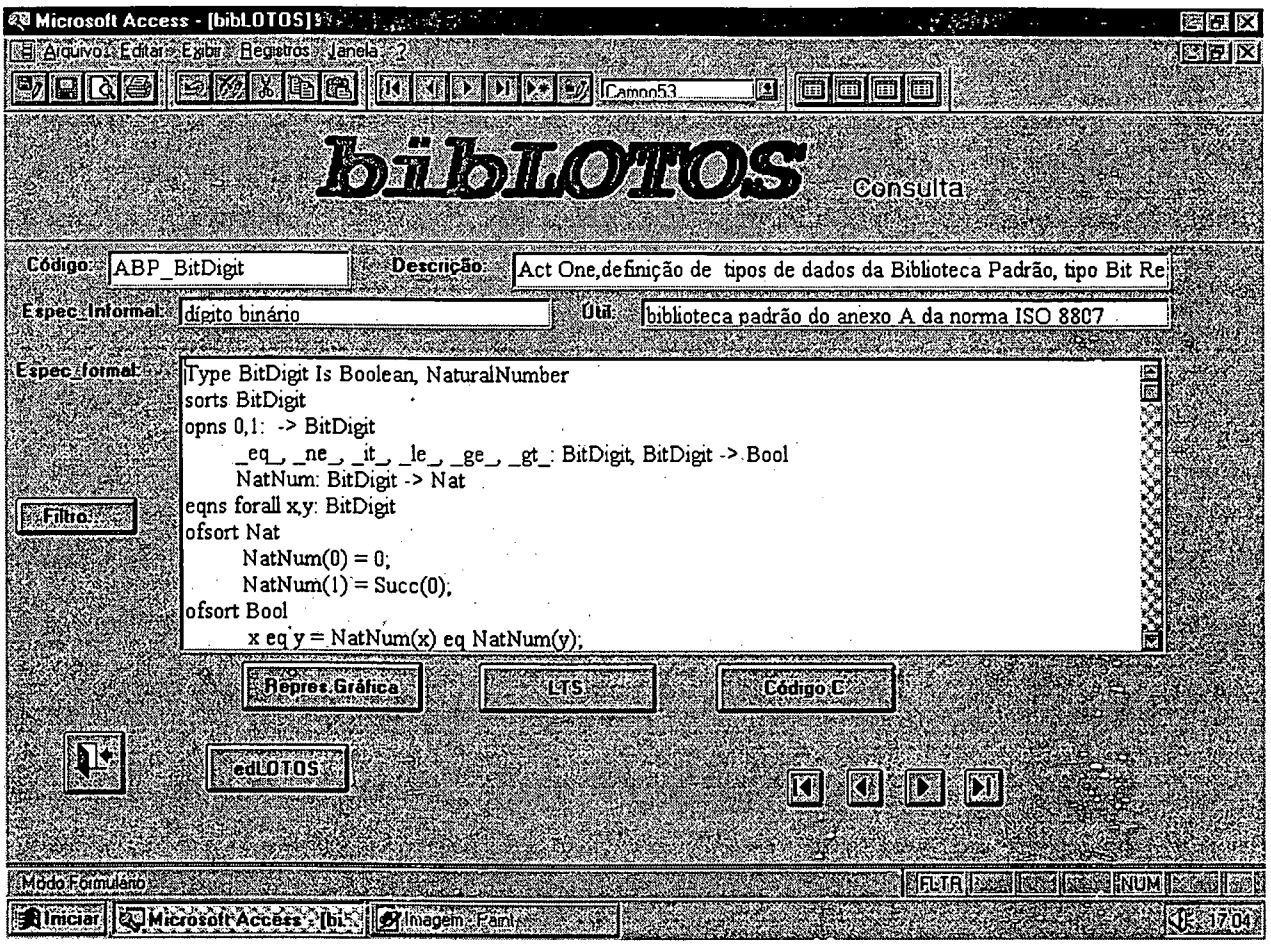


Figura 4.10 - Tela de consulta.

Se o projetista deseja transferir a construção predefinida da bibLOTOS para o editor de especificações edLOTOS, então ele executa os seguintes passos:

1) seleciona a construção predefinida, clicando com o *mouse* sobre o nome do campo **Espec_formal** - na tela de consulta;

2) pressiona o botão  da barra de ferramentas (ou opção Editar/Copiar da barra de menu), transferindo a construção selecionada no passo 1 para a área de armazenamento do *Windows*;

3) pressiona o botão automático  , o qual abrirá o editor de especificações (sobrepondo-o à tela de consulta);

4) na barra de menu escolhe a opção **Editar/Colar**, inserindo assim no editor a construção predefinida selecionada no passo 1.

Veja o resultado desta seqüência de passos na Figura 4.11.

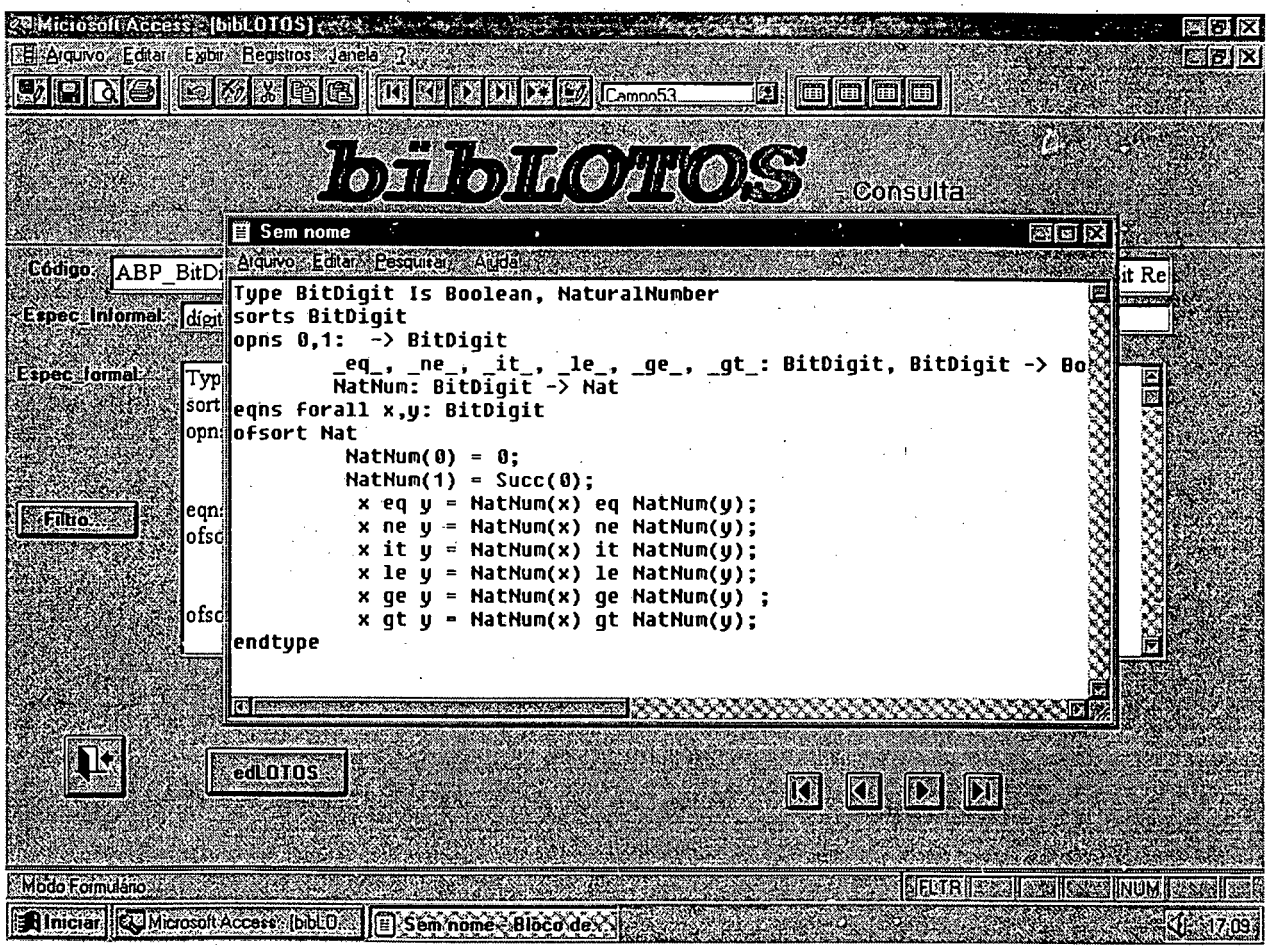


Figura 4.11 - Tela de consulta/Tela do edLOTOS.

Após inserir a construção predefinida no edLOTOS, o projetista pode fazer as modificações de acordo com as necessidades do seu projeto. Caso necessite de outra construção, o projetista minimiza (botão ▣ da janela do editor) o edLOTOS e repete os passos descritos acima.

Através de botões automáticos o projetista pode acessar os formulários (representados por janelas) que contêm informações adicionais referentes a cada construção predefinida. Os botões representação gráfica, LTS e código C auxiliam o projetista na tomada de decisão acerca da construção predefinida ideal.

A tela de cadastro tem como finalidade permitir aos usuários da bibLOTOS inserir novas construções predefinidas, aumentando assim o acervo da biblioteca, ou alterar os dados das construções já armazenadas. Nesta tela são apresentados os mesmos campos da tela de consulta. Veja a Figura 4.12.

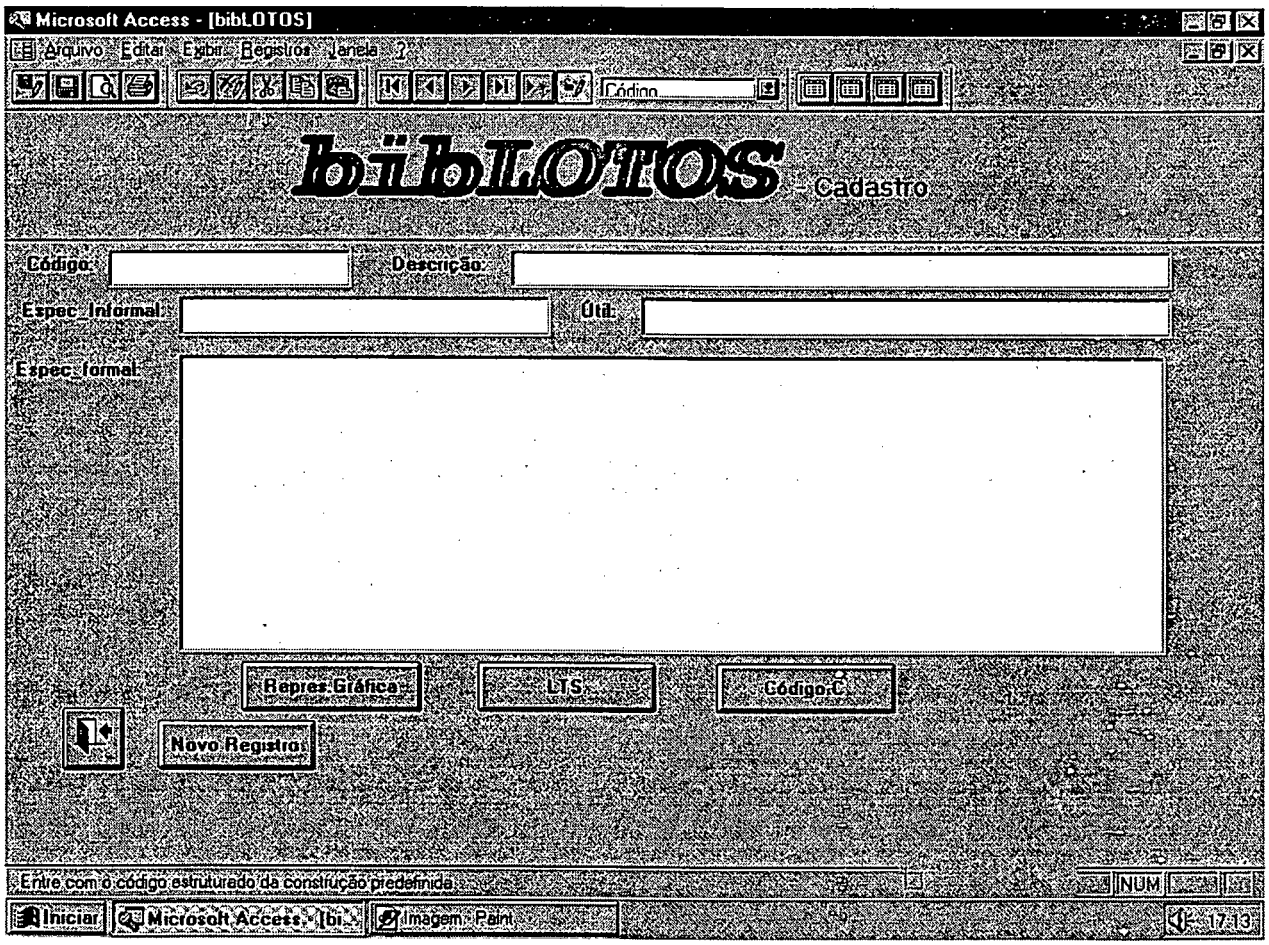


Figura 4.12 - Tela de cadastro.

O botão automático **Novo Registro** abre um registro na biblioteca, possibilitando a inserção de uma nova construção predefinida e de seus respectivos dados.

Após efetuar o cadastro de uma nova construção predefinida (tela de cadastro), ou modificar uma já existente (tela de consulta), o projetista deve atualizar o banco de dados. Este procedimento é realizado através do botão Salvar -  - da barra de ferramentas personalizada para a bibLOTOS.

No próximo capítulo é apresentado o método proposto para o desenvolvimento de sistemas baseado no paradigma de reutilização.

5. Método Proposto para o Desenvolvimento de Sistemas

"Despite dramatic increases in network and host performance, it remains difficult to design, implement, and reuse communication software for complex distributed systems. Examples of these systems include global personal communication systems, network management platforms, enterprise medical imaging systems, and real time market data monitoring and analysis system" [Schm 95].

O desenvolvimento de sistemas concorrentes ou distribuídos complexos é uma tarefa difícil. Para enfrentar tal complexidade a descrição de tais sistemas pode ser realizada com a redução progressiva dos níveis de abstração. Além disso, o processo de desenvolvimento desses sistemas pode ser beneficiado com a adoção de procedimentos sistemáticos baseados na utilização de técnicas formais.

Neste capítulo é proposto um método para o desenvolvimento de sistemas concorrentes ou distribuídos, que considera, basicamente, a técnica de descrição formal LOTOS. Com tal proposta busca-se definir um processo de desenvolvimento que seja sistemático e rigoroso. O interesse principal é introduzir a utilização de construções predefinidas LOTOS na especificação de sistemas, adotando-se uma abordagem de refinamentos sucessivos.

5.1. O Processo de desenvolvimento de *Software*

O processo de desenvolvimento de sistemas é um conjunto de atividades relacionadas. A expressão desenvolvimento de sistemas incorpora todas as atividades relacionadas ao sistema e ao contexto do qual ele é parte, incluindo desde a definição do problema até a manutenção do sistema [Luce 87].

Para se estabelecer um processo de desenvolvimento, os princípios da engenharia de *software* devem ser adotados para se obter a garantia da qualidade. Tal engenharia abrange um conjunto de três elementos fundamentais: métodos, ferramentas e procedimentos - que dão ao desenvolvedor o controle do processo de desenvolvimento e oferecem a base para a obtenção de sistemas de alta

qualidade. Os métodos da engenharia de *software* proporcionam os detalhes de "como fazer" para construir o sistema, as ferramentas proporcionam apoio automatizado ou semi-automatizado aos métodos, e os procedimenos definem a seqüência em que os métodos são aplicados [Pres 95].

A engenharia de *software* compreende um conjunto de etapas (repetitivas ou evolutivas) que envolvem métodos, ferramentas e os procedimentos mencionados acima. Essas etapas constituem o paradigma de engenharia de *software* [Pres 96].

Para estabelecer um processo de desenvolvimento de sistemas, inicialmente pode-se estabelecer o paradigma de engenharia de *software* a ser utilizado. Quatro paradigmas têm sido amplamente citados: o ciclo de vida clássico, a prototipação, o modelo espiral e as técnicas de quarta geração [Luce 87].

O paradigma mais adequado ao processo de desenvolvimento apresentado nesta dissertação é o ciclo de vida clássico, visto que as atividades para este paradigma são, pela ordem:

- Análise dos requisitos do contexto;
- Análise dos requisitos do sistema;
- projeto;
- codificação;
- teste;
- manutenção.

A escolha de um paradigma de engenharia de *software* deve ter como base a natureza do projeto e da aplicação, os métodos e as ferramentas a serem usados, o controle do processo de desenvolvimento e o produto final que precisa ser entregue [Pres 95].

5.1.1 Reutilizabilidade (Aspectos de Qualidade)

A reutilizabilidade de um produto é um aspecto de qualidade que pode ser observado pela facidade de uso desse produto. A presença ou ausência de reutilizabilidade em um produto pode ser detectada pelos usuários do produto [Meye 88]. Por sua vez, uma especificação que pode ser reutilizada, no todo ou em parte, para se obter uma nova especificação é uma especificação que goza da propriedade da reutilização [Sobr 96].

LOTOS é uma técnica de descrição formal que se pretende empregar juntamente com um método que reutilize especificações ao longo do desenvolvimento de sistemas. Tal método baseia-se na hipótese de que existem sistemas com aspectos similares, o que sugere a reutilização de componentes de especificações. Estes componentes são denominados construções predefinidas.

As construções predefinidas são classificadas segundo as suas características. Elas são classificadas em grupos, podendo ser recuperadas separadamente para reutilização. A reutilização de construções predefinidas, durante o refinamento de um sistema, é apropriada para economia de tempo, durante a fase de edição.

Projetistas admitem que a reutilização pode melhorar a produtividade do processo de desenvolvimento e a qualidade dos produtos obtidos. As abordagens propostas para a reutilização de *software* concentram-se em dois aspectos: criação de componentes reutilizáveis e adoção de um método de reutilização [ChCh 96]. A criação de componentes é enfocada no Capítulo 3 (Construções Predefinidas), onde diversas construções predefinidas são apresentadas e posteriormente armazenadas na biblioteca bibLOTOS. O método de reutilização proposto neste

capítulo é, por sua vez, ilustrado no Capítulo 6 (Exemplo de Aplicação da bibLOTOS e do Método Associado), onde as construções predefinidas são empregadas de modo sistemático.

Muitas técnicas para reutilização de código de programação têm sido desenvolvidas, como por exemplo a reutilização de componentes em C++. Fazendo uso dessas técnicas, o projeto e a especificação final de um sistema pode estar disponível antes do código final ser produzido. Deste modo o tempo de implementação pode se reduzir, o mesmo acontecendo com o tempo de projeto e análise. Para aumentar o poder da reutilização, alguns pesquisadores [Luba 89] [KaRi 87] [ChCh 94] [KhLi 94] têm desenvolvido técnicas para reutilização de projetos e especificações. Técnicas para reutilização de especificações, de projetos e de código de programa podem ser integradas com o objetivo de apoiar um paradigma de desenvolvimento baseado na reutilização [ChCh 96].

A reutilização de construções predefinidas é, portanto, uma abordagem de projeto promissora. Desse modo, a reutilização de especificações LOTOS no contexto de gerência de redes pode ser investigada. Especificações LOTOS reutilizáveis podem ser elaboradas, classificadas, armazenadas e recuperadas através de um código estruturado e posteriormente empregadas de acordo com um procedimento sistemático, por exemplo, no projeto de sistemas para a gerência de redes.

5.1.2 Considerações Básicas

No processo de desenvolvimento de sistemas, a reutilização de construções predefinidas tem como considerações básicas:

1. Dois ou mais sistemas com **arquiteturas** similares podem ser representados através da combinação dos mesmos componentes, ou de componentes semelhantes. Exemplo: no projeto de sistemas distribuídos pode-se adotar uma estrutura arquitetônica modular onde se destacam os aspectos locais (relativos a cada sítio) e os aspectos fim-a-fim (relativos à comunicação entre os sítios) [Riso 91].

2. Dois ou mais sistemas com **comportamentos** similares podem ser representados através dos mesmos componentes ou de componentes semelhantes. As especificações desses sistemas primeiramente fixam decisões relativas a funções e dados. Especificações cujas funções sejam similares são reutilizáveis em um sistema em desenvolvimento. Se a especificação reutilizada e a especificação do sistema em desenvolvimento processarem diferentes dados, então os dados da especificação reutilizada podem ser modificados. Desde que a execução das funções exibam comportamentos semelhantes, as especificações correspondentes podem ser elaboradas com o emprego dos mesmos processos. Exemplo: no gerenciamento de redes, quando os dados são coletados de diferentes MIBs (MIB, RMON MIB, MIB II), o comportamento pode ser reutilizado e a estrutura dos dados modificados conforme o tipo de MIB.

3. Métodos e técnicas de desenvolvimento estão associadas ao **ambiente** no qual são empregados. O uso de ferramentas automatizadas de apoio à edição de especificações induz o projetista à adoção das linhas de raciocínio embutidas nas ferramentas. Exemplo: emprego de sistemas operacionais e editores de texto baseados em janelas.

4. A reutilização pode se tornar especialmente eficiente quando aplicada a **domínios de aplicações** restritos, sobre os quais já exista conhecimento sedimentado, disponível de forma

estruturada e acessível [ZiOl 94]. Exemplo: neste trabalho o domínio de aplicação proposto é a gerência de redes. O conhecimento nesse domínio vem sendo sedimentado pela equipe do LRG, principalmente através das atividades do projeto PLAGERE.

O método proposto contempla as considerações básicas levantadas acima, tendo como domínio de aplicação a gerência de redes de computadores e telecomunicações, em particular, a gerência proativa dessas redes.

5.2 O Processo de Desenvolvimento de Especificações

Para desenvolver a especificação de um sistema, com a utilização de construções predefinidas reutilizáveis, um projetista pode seguir as indicações apresentadas abaixo. Essas indicações tratam do método, das ferramentas e dos procedimentos propostos nesta dissertação para o desenvolvimento de especificações LOTOS.

1. Inicialmente ele define como o sistema se comporta e procura representar este comportamento através da composição de construções predefinidas. Para tal, ele pode localizar as construções predefinidas de seu interesse através do código estruturado de armazenamento dessas construções. A pesquisa das construções predefinidas, pelo código estruturado, dá-se através de um filtro, com opções para acesso às construções LOTOS Básico, ACT ONE ou LOTOS Completo.
2. O projetista examina a construção predefinida e pode selecioná-la para reutilização. Os diversos campos da ferramenta bibLOTOS, tais como especificação informal, representação gráfica e LTS (Sistema de Transições Rotuladas), por exemplo, facilitam a seleção da

construção que melhor satisfaça ao projetista durante a fase de desenvolvimento de um sistema.

3. O projetista seleciona as construções predefinidas necessárias para a montagem da especificação em desenvolvimento. A transferência da construção predefinida da bibLOTOS para a especificação de um sistema dá-se através de um botão automático (uma função macro, representando as opções *copy/paste* no menu Editar).
4. O editor de especificações (edLOTOS), do ambiente LOWE, permite que a construção predefinida recuperada seja modificada adequadamente até que ela passe a representar fielmente a intenção do projetista.

De acordo com as indicações esquematizadas acima, as principais etapas do método são: classificação, armazenamento e recuperação de construções predefinidas, exame de especificações (com o emprego da ferramenta bibLOTOS) e modificação e composição de especificações (com utilização da ferramenta edLOTOS). A Figura 5.1 representa o processo de desenvolvimento da especificação de um sistema com uso das construções predefinidas armazemadas na bibLOTOS.

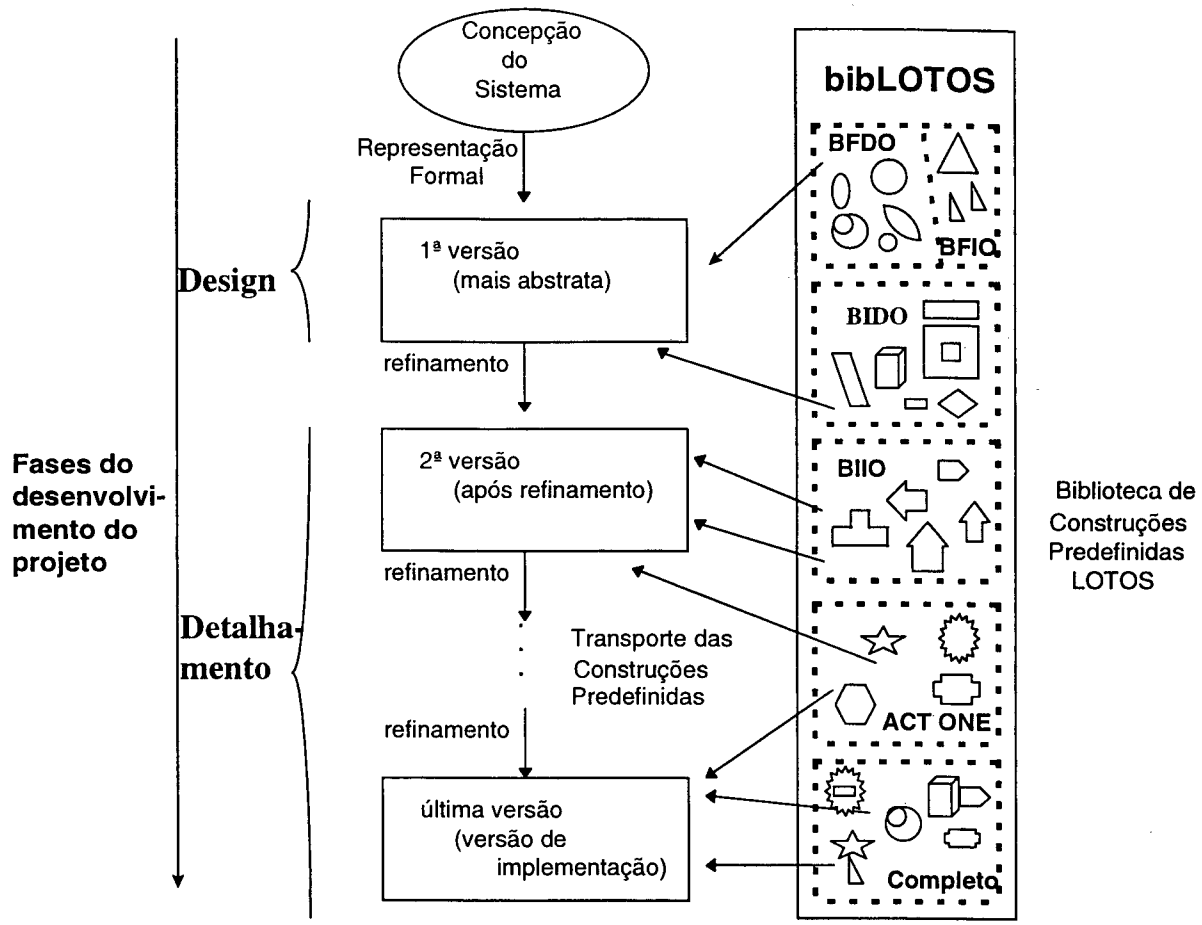


Figura 5.1 - Visão esquemática do processo de desenvolvimento de sistemas com o uso de construções predefinidas armazenadas na bibLOTOS.

O método proposto adota uma abordagem de projeto *top-down*. O procedimento adotado sugere que, partindo de uma especificação em alto nível de abstração, novas especificações, cada vez mais refinadas, vão sendo obtidas à medida que as decisões de projeto são incorporadas à especificação. Cada nova versão é comparada com a versão anterior. A nova versão é considerada correta se for equivalente à versão anterior. A prova de equivalência é realizada com o auxílio de ferramentas automáticas. Veja a Figura 5.2.

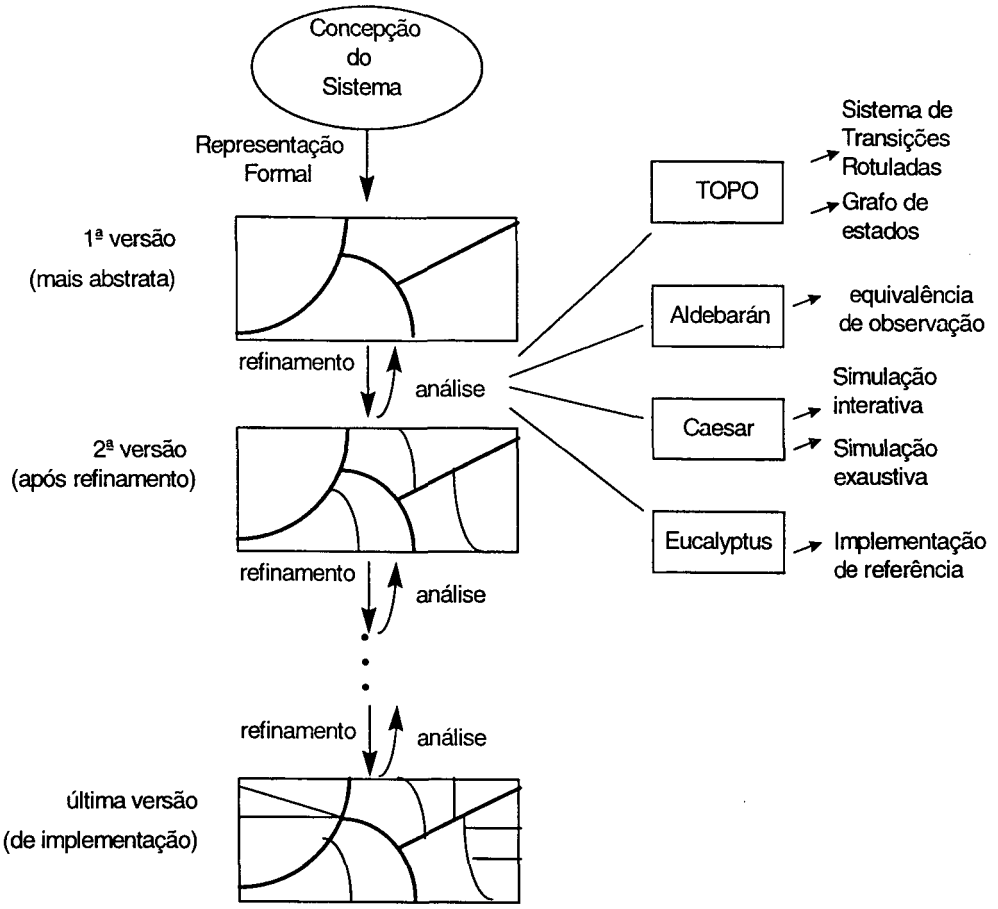


Figura 5.2 - Visão esquemática do processo de análise das sucessivas versões de sistema em desenvolvimento com o uso de ferramentas automáticas de auxílio ao projeto (TOPO, *Aldébaran*, *Cæsar* e *Eucaliptus*).

Implementações de referência (que podem ser utilizadas como ponto de partida para a criação de programas eficientes) são geradas, automaticamente, a partir de cada versão da especificação do sistema em desenvolvimento. Em particular, a última versão (denominada versão de implementação ou de realização do sistema) permite obter a implementação de referência mais completa.

A reutilização dos componentes é o ponto alto da abordagem que faz uso de construções predefinidas. Projetistas, contudo, podem ter dificuldades no desenvolvimento e na manutenção de bibliotecas de construções predefinidas. Métodos automáticos para transformar

especificações genéricas em composições de construções predefinidas não estão ainda à disposição, e assim um tempo considerável pode ser consumido em tarefas realizadas manualmente, em alguns casos.

5.3 Ambiente de Suporte (ferramentas)

Para dar suporte ao método proposto um ambiente de desenvolvimento foi concebido e implementado. O ambiente LOWE (*LOTOS Windows Environment*), que suporta o método proposto, está implementado em ambiente compatível com o IBM PC. Ele é composto pelas seguintes ferramentas:

1. **Editor de especificações (edLOTOS):** que é utilizado para editar especificações que são classificadas e armazenadas em um repositório para reutilização.

2. **Biblioteca de especificações (bibLOTOS):** que armazena registros com construções predefinidas e uma série de outros campos. Ela é indexada por um código estruturado para facilitar a recuperação de especificações. Estes códigos estruturados são visualizados em uma tabela, numa caixa de diálogo do *Windows*.

3. **Transformação de especificações (TranSP):** esta ferramenta permite transformar automaticamente especificações formais de serviço em especificações formais de protocolo. O formalismo utilizado é LOTOS Básico. As transformações de especificações são realizadas através de algoritmos desenvolvidos com base na relação entre os conceitos de serviço e protocolo. A implementação desses algoritmos é feita em linguagem C++ e constitui a ferramenta TranSP [Toni 96].

A integração da bibLOTOS com as ferramentas do ambiente LOWE, durante as fases de descrição das especificações, ocorre através da tela de consulta desta biblioteca, com opções para chamada ao editor das especificações.

5.3.1 Configuração Mínima de *Hardware* e *Software*

A configuração mínina de *Hardware* e *Software* exigida para a utilização das ferramentas do ambiente LOWE é:

- Hardware mínimo: microcomputador IBM-PC ou compatível, com um processador 80386SX ou superior, um disco rígido de 20 *megabytes* de espaço livre, com 8 MB de memória RAM, *mouse* e monitor colorido 17".

- *Software*: Windows 3.1 ou Windows 95.

6. Exemplo de Aplicação da bibLOTOS e do Método Associado

"Formal project specifications are hierarchical, tracing the requirements and development phases of the life cycle" [Evan 89].

Neste capítulo apresenta-se um exemplo que ilustra o emprego de construções predefinidas no projeto de um sistema de gerência de redes. Em tal exemplo considera-se a especificação de um agente proativo, que pode ser utilizado para o gerenciamento do tráfego de pacotes nas redes de comunicação.

6.1 Gerência proativa de redes

No âmbito das pesquisas realizadas no LRG da UFSC vêm sendo investigados vários aspectos relacionados à gerência proativa de redes. Esse tipo de gerência tem como objetivo prever os problemas que podem ocorrer na rede e evitar que eles degradem os serviços oferecidos aos usuários. Portanto, a gerência proativa deve ser capaz de localizar os indícios de problemas antes que tais problemas efetivamente aconteçam, sejam eles de contabilização, de segurança, de falha, de configuração ou de desempenho.

A aplicação da gerência proativa se faz através de um sistema complexo, onde muitos elementos específicos são combinados, para realizar esse tipo de gerenciamento. A complexidade da gerência proativa de redes sugere o uso de rigorosas abordagens de projeto. A engenharia destes sistemas pode ser feita com as vantagens oferecidas pelo uso de Técnicas de Descrição Formal (TDFs).

Diversas abordagens podem ser adotadas para o projeto e a implementação de sistemas para gerência de redes. A abordagem que faz uso de construções predefinidas é o alvo deste trabalho.

Para dar suporte a tal abordagem uma biblioteca de construções predefinidas, a bibLOTOS, foi implementada, e a ela está associado um método que orienta a reutilização de construções predefinidas. O conjunto de informações associadas às construções predefinidas presentes na bibLOTOS facilita as atividades de projeto.

6.2 Descrição de um agente proativo de redes

Nas seções seguintes desenvolve-se um estudo visando o desenvolvimento de uma aplicação para o gerenciamento de redes, através da especificação formal, com o uso das construções predefinidas da bibLOTOS e com o uso do método proposto. Essa aplicação deve oferecer serviços de gerência proativa de redes. A aplicação não é completamente especificada. O componente da gerência proativa cujo comportamento é especificado formalmente a seguir é o Módulo de Gerência Proativa.

A aplicação especificada adota uma abordagem proativa para realizar a tarefa de monitoramento do tráfego de mensagens de pacotes em uma rede de comunicação. A especificação formal desta aplicação faz uso das construções predefinidas presentes na bibLOTOS.

6.2.1 Descrição informal

O agente proativo PROAG é responsável por monitorar e analisar o comportamento da rede. Ele realiza operações (representadas por eventos na porta *operat*) sobre os objetos gerenciados da MIB (MIB II e/ou RMON MIB) e compara os valores obtidos com os valores armazenados na *BASELINE* (o acesso aos valores armazenados na *BASELINE* é representado através de eventos na porta *baseline_data*).

Em alto nível de abstração o agente proativo (denominado PROAG) pode ser visto como um sistema onde somente as portas de entrada e as portas de saída de dados são observáveis. Por não se considerar a estrutura interna do sistema é que esta estrutura é denominada de caixa preta. Veja a Figura 6.1.

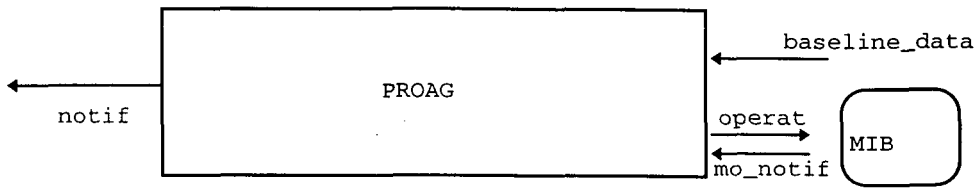


Figura 6.1 - Representação gráfica do agente proativo em alto nível de abstração.

O envio de notificações para o gerente é representado por eventos na porta *notif*. Eventualmente, o agente proativo PROAG pode receber notificações provenientes da MIB (neste caso ocorre um evento na porta *mo_notif).* Este é o caso que se dá, por exemplo, quando o valor de um parâmetro de utilização de um recurso da rede excede um limite preestabelecido.

6.2.2 Descrição formal em alto nível de abstração

Considerando a representação do agente proativo PROAG, conforme a Figura 6.1, a especificação formal LOTOS correspondente, em alto nível de abstração, possui um único processo: o processo PROA.

Utilizando o método proposto deve-se, inicialmente, definir quais as características da construção predefinida a ser recolhida da bibLOTOS, a fim de especificar o agente proativo PROAG. A construção predefinida, que possui um único processo e quatro portas de comunicação com o

ambiente externo, deve satisfazer as seguintes características: 1) fazer uso somente da parte de comportamento, ou seja, LOTOS Básico (B); 2) comportar-se de forma recursiva (I); 3) possuir opções de escolha indeterminística entre expressões de comportamento (I); e 4) conter eventos observáveis (O).

Segundo estas características (utilizando-se, também, da representação gráfica da construção predefinida armazenada na bibLOTOS), neste nível de abstração, uma construção do tipo BIIO pode ser capturada da bibLOTOS. Veja abaixo a construção BIIO_BASICO4[a,b,c,d] que melhor representa as características da especificação do agente proativo PROAG em alto nível de abstração.

```
specification BIIO_BASICO4[a,b,c,d] : noexit : =
behaviour
        BIIO_BASICO4[a,b,c,d]
where
        process BIIO_BASICO4[a,b,c,d]
        ...
        endproc
endspec
```

Na bibLOTOS, através de um botão automático, o projetista recolhe a construção predefinida que é do seu interesse e a insere no editor de especificações LOTOS (edLOTOS), onde a construção predefinida é modificada. Veja abaixo as alterações realizadas na construção predefinida BIIO_BASICO4[a,b,c,d], para satisfazer as necessidades do projetista.

```
specification PROAG[notif,baseline_data,mo_notif,operat] : noexit
behaviour (*o comportamento do agente proativo é definido pelo processo PROA *)
   PROA[notif,baseline_data,mo_notif,operat]
where
   process PROA[notif,baseline_data,mo_notif,operat]:noexit :=
        :
   endproc
endspec
```

Visto como uma caixa preta, o processo PROA possui quatro portas de comunicação com o ambiente externo (são elas: *notif,baseline_data,mo_notif,operat*). O comportamento de PROA é descrito informalmente como segue.

No processo PROA tem-se, inicialmente, uma escolha indeterminística (representada pelo operador [ ]) na qual, ou o agente proativo solicita uma operação (por exemplo, um SET) à MIB através da porta *operat*, ou o agente proativo recebe da MIB uma notificação (por exemplo, um TRAP) através da porta *mo_notif*, quando necessário.

Após a ocorrência de uma das alternativas dessa escolha o agente obtém informações da *BASELINE*, através de um evento na porta *baseline_data*. Ocorre então uma nova escolha indeterminística, onde após uma tomada de decisão (representada por um evento interno *i*), ou o processo PROA é chamado recursivamente, ou é enviada uma notificação (neste caso ocorre um evento na porta *notif*) para um gerente, antes que o processo PROA seja chamado recursivamente. Observe a especificação apresentada abaixo.

```
process PROA[notif,baseline_data,mo_notif,operat] : noexit :=
   operat;                                    (*realiza operações sobre os objetos   *)
      baseline_data;                          (*e faz comparações com a BASELINE*)
       (i; PROA[notif,baseline_data,mo_notif,operat]      (*decide reinicializar*)
     [] i;notif; PROA[notif,baseline_data,mo_notif,operat]) (*decide emitir notificações*)
[] mo_notif;                                  (*recebe informações da MIB e faz *)
      baseline_data;                          (* comparações com a BASELINE  *)
      (i; PROA[notif,baseline_data,mo_notif,operat]          (*decide reinicializar*)
     []i; notif; PROA[notif,baseline_data,mo_notif,operat]) (*decide emitir notificações*)
endproc
```

Observando o comportamento descrito pelo processo PROA podem ser identificadas duas partes muito semelhantes. Essas partes estão separadas pelo primeiro nível de escolha indeterminística (separação esta realizada com a utilização do operador []).

A primeira parte do processo PROA é:

operat;baseline_data;
(i; PROA[*notif,baseline_data,mo_notif,operat*]
[] i;*notif*; PROA[*notif,baseline_data,mo_notif,operat*])

A segunda parte do processo PROA é:

mo_notif;baseline_data;
(i; PROA[*notif,baseline_data,mo_notif,operat*]
[] i;*notif*; PROA[*notif,baseline_data,mo_notif,operat*])

Pode-se observar nas duas partes do processo PROA, que após executar dois eventos observáveis, ou ele decide reinicializar imediatamente ou decide emitir uma notificação antes de reinicializar.

Considerando a semelhança de comportamentos, cada uma dessas partes pode ser especificada com a utilização de uma instância de uma construção predefinida (neste caso, a construção predefinida BIII_PREDEFINIDO) armazenada na bibLOTOS. Veja abaixo esta construção predefinida.

process BIII_PREDEFINIDO[a,b,c,d] : **noexit** : =
b;
(i; BIII_PREDEFINIDO[a,b,c,d]
[]
i;a; BIII_PREDEFINIDO[a,b,c,d])
endproc

Neste caso a adoção de construções predefinidas permitiu dar a especificação do agente uma feição padronizada, inclusive tornando mais clara a interpretação da estrutura da especificação.

specification PROAG_1[*notif,baseline_data,mo_notif,operat*] : **noexit**
behaviour
PROA[*notif,baseline_data,mo_notif,operat*]

```
where
  process PROA[notif,baseline_data,mo_notif,operat] : noexit : =
    operat;BIII_PREDEFINIDO[notif,baseline_data,mo_notif,operat]
     []
    mo_notif;BIII_PREDEFINIDO[notif,baseline_data,mo_notif,operat]
    where
      process BIII_PREDEFINIDO[notif,baseline_data,mo_notif,operat] : noexit : =
        baseline_data;
        (i; PROA[notif,baseline_data,mo_notif,operat]
        []
         i;notif; PROA[notif,baseline_data,mo_notif,operat])
      endproc
  endproc
endspec
```

A especificação do agente proativo PROAG, após a composição, pode ser vista acima. Pode-se provar que as especificações PROAG e PROAG_1 (duas versões diferentes do mesmo sistema) são equivalentes.

6.2.3 Descrição detalhada do agente

Ao refinar o agente proativo pode-se considerar uma estrutura interna com dois componentes: o *REMOTE_MONITOR* e o *VERIFY*. A combinação desses componentes define a estrutura do agente proativo.

O *REMOTE_MONITOR* (Monitor de Rede) coleta informações de objetos gerenciados das MIB*s* tais como a *Remote Monitoring Management Information Base* (RMON MIB) e a *Management Information Base II* (MIB II).

O *VERIFY* (Serviço de Verificação) é responsável por verificar as tendências de degradação da rede. Isto é feito por meio de comparações entre os valores armazenados na *BASELINE* e os

valores coletados, constantemente, pelo *REMOTE_MONITOR*. As notificações são enviadas para a um gerente através de eventos na porta *notif*. Veja a Figura 6.2.

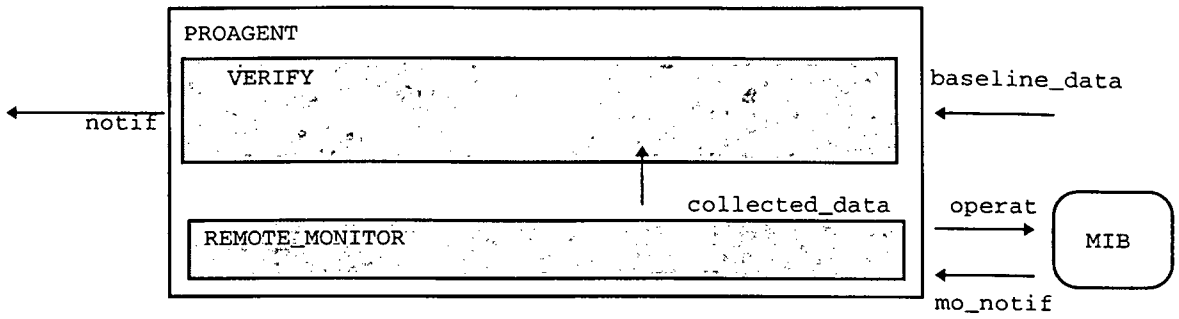


Figura 6.2 - Representação gráfica do agente proativo.

Considerando a Figura 6.2, a estrutura interna do agente proativo PROAGENT pode ser especificada como segue.

specification PROAGENT[*notif,baseline_data,operat,mo_notif*]: **noexit**
behaviour
hide *collected_data* **in**
VERIFY[*notif,baseline_data,collected_data*]
|[*collected_data*]|
REMOTE_MONITOR[*operat,mo_notif,collected_data*]
where
process VERIFY[*notif,baseline_data,collected_data*]: **noexit** :=
...
endproc
process REMOTE_MONITOR[*operat,mo_notif,collected_data*] : **noexit** : =
...
endproc
endspec

Esta especificação pode ser obtida a partir da construção predefinida BIII_PROV_COM, armazenada na bibLOTOS.

A especificação refinada que apresenta a estrutura do agente proativo permanece com quatro portas de comunicação com o ambiente externo (*notif, baseline_data, mo_notif, operat*). Internamente ao PROAGENT são definidos dois processos: REMOTE_MONITOR e VERIFY.

Eles são combinados com a utilização do operador de composição geral (operador |[ ]|). Os processos REMOTE_MONITOR e VERIFY, compartilham a porta *collected_data* que é ocultada com a utilização do operador *hide...in....*

O processo REMOTE_MONITOR comunica-se com a MIB através das portas *operat* e *mo_notif.* No comportamento deste processo tem-se uma escolha indeterminística. O agente proativo solicita uma operação (por exemplo, um *SET*) à MIB através da porta *operat,* coleta os dados e sincroniza-se com o processo VERIFY na porta *collected_data.* O processo REMOTE_MONITOR é chamado recursivamente. Caso necessário, a MIB envia uma notificação (por exemplo, um TRAP) ao REMOTE_MONITOR. O recebimento de tal notificação por parte do processo REMOTE_MONITOR é representado por um evento na porta *mo_notif.*

Ao refinar o componente VERIFY pode-se definir a existência dos processos COMPARE e CALCULATE, como apresentado na Figura 6.3.

O processo *CALCULATE* recolhe os dados da porta *collected_data,* realiza os cálculos de variável para enviá-los ao processo *COMPARE* pela porta *calculated_data.* Este processo é chamado recursivamente.

O processo *COMPARE* recebe os dados da porta *calculated_data,* comunica-se com a base de dados através da porta *baseline_data,* a fim de coletar os dados sobre o funcionamento da rede, e realiza uma escolha indeterminística. Após a ocorrência de um evento interno `i` ou é enviada uma notificação para o gerente (na porta *notif),* ou os dados são calculados e comparados novamente, para então chamar o processo *COMPARE* recursivamente.

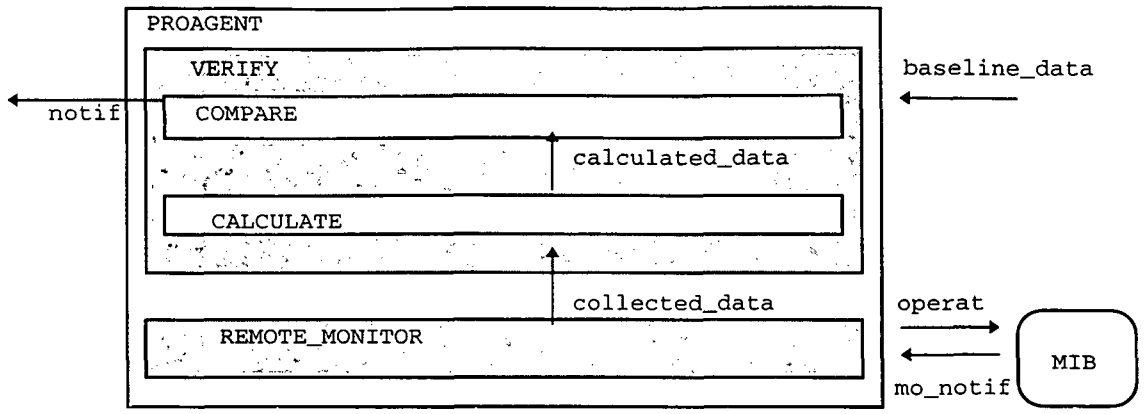


Figura 6.3 - Representação gráfica refinada do agente proativo .

Uma especificação mais refinada do agente proativo é apresentada a seguir. Essa especificação refinada é identificada como PROAGENT.

specification PROAGENT[*notif,baseline_data,operat,mo_notif*]: **noexit**
behaviour
hide *collected_data* **in**
VERIFY[*notif,baseline_data,collected_data*]
|[*collected_data*]|
REMOTE_MONITOR[*operat,mo_notif,collected_data*]
where
process VERIFY[*notif,baseline_data,collected_data*]: **noexit** :=
hide *calculated_data* **in**
CALCULATE[*collected_data,calculated_data*]
|[*calculated_data*]|
COMPARE[*notif,baseline_data,calculated_data*]
where
process CALCULATE[*collected_data,calculated_data*] : **noexit** : =
collected_data;calculated_data;
calculated_data;collected_data; CALCULATE[*collected_data,calculated_data*]
endproc
process COMPARE[*notif,baseline_data,calculated_data*] : **noexit** : =
calculated_data;baseline_data;
(i;*calculated_data*;COMPARE[*notif,baseline_data,calculated_data*]
[ ] i;*notif;calculated_data*; COMPARE[*notif,baseline_data,calculated_data*])
endproc
endproc
process REMOTE_MONITOR[*operat,mo_notif,collected_data*] : **noexit** : =
operat;collected_data;collected_data;REMOTE_MONITOR[*operat,mo_notif,collected_data*]
[ ] *mo_notif;collected_data;collected_data;*
REMOTE_MONITOR[*operat,mo_notif,collected_data*]
endproc

endspec

Os processos COMPARE, CALCULATE e REMOTE_MONITOR são instâncias de construções predefinidas (processos predefinidos) armazenados na bibLOTOS.

6.3 Análise automática das especificações

Duas ferramentas são utilizadas no trabalho de análise, validação e tradução de especificações: (1) CADP - *Cæsar Aldébaran Distribution Package* (caesar@imag.fr) [Gara 94] e (2) *TOPO* (topo@dit.upm.es). As ferramentas que compõem o CADP estão incluídas no ambiente gráfico Eucalyptus.

6.3.1 Análise Sintática e Semântica

Inicialmente, a análise sintática e semântica do agente é realizada com a utilização da ferramenta Cæsar. Essa análise pode ser feita de duas maneiras: a partir da linha de comando Unix ou a partir do ambiente gráfico Eucalyptus. De ambas as maneiras os passos executados nessa análise são os apresentados na Figura 6.4.

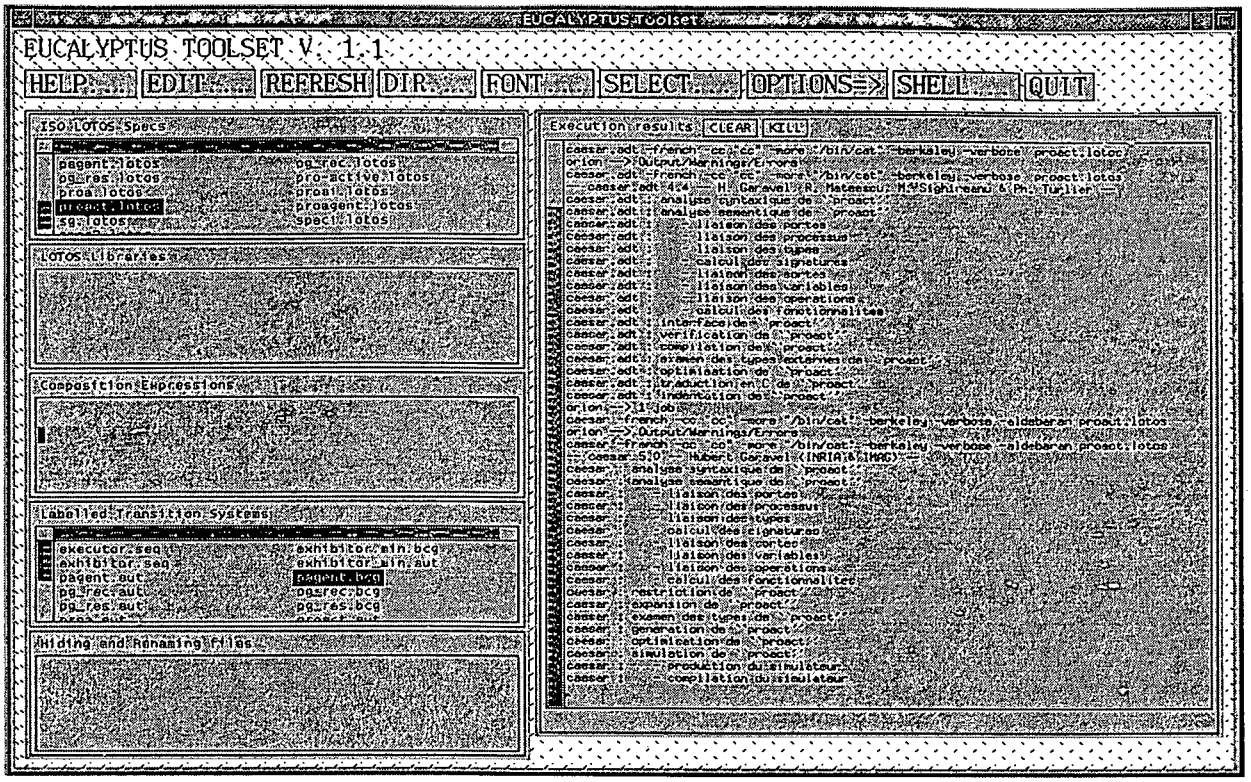


Figura 6.4 - Procedimento de análise do agente proativo no ambiente Eucalyptus.

6.3.2 Simulação do Agente Proativo

A simulação contribui para a validação de especificações. Nesta seção são apresentadas duas espécies de simulação: a) A primeira é a simulação exaustiva onde todos os estados e transições alcançáveis são representados em um grafo de estados; b) A outra é a simulação interativa na qual o projetista pode escolher um caminho a percorrer.

a. Simulação Exaustiva

Durante a compilação uma rede de Petri é criada. Esta rede (arquivo `.net`) é apresentada a seguir.

```
proagent.net

reseau pour la specification
``proagent''

nombre de variables: 0
nombre d'unites: 4
nombre de places: 12
nombre de transitions: 13
nombre de marques: 3

liste des variables

liste des registres

liste des constantes

liste des unites

unite 0
    place unique initiale: 0
       sous-unites:
unite 1
       places: 1 ... 3
```

```
      place initiale: 1
unite 2
      places: 4 ... 9
      place initiale: 4
unite 3
      places: 10 ... 11
      place initiale: 10

liste des transitions ``epsilon''

    transition 0
       places d'entree: 0
       places de sortie: 1 4 10

liste des transitions visibles

    transition 1
       places d'entree: 1
       places de sortie: 2
       porte: MO_NOTIF.0.0[1]
    transition 2
       places d'entree: 1
       places de sortie: 3
       porte: OPERAT.0.0[1]
    transition 3
       places d'entree: 9
       places de sortie: 8
       porte: BASELINE_DATA.0.0[1]
    transition 4
       places d'entree: 8
       places de sortie: 4
       porte: NOTIF.0.0[1]
    transition 5
       places d'entree: 7
       places de sortie: 6
       porte: BASELINE_DATA.0.0[1]
    transition 6
       places d'entree: 6
       places de sortie: 4
       porte: REQ.0.0[1]
    transition 7
       places d'entree: 5
       places de sortie: 4
       porte: BASELINE_DATA.0.0[1]

liste des transitions ``tau''

    transition 8
       places d'entree: 4 11
       places de sortie: 5 10
       porte: CALCULATED_DATA.0.1[3]
    transition 9
       places d'entree: 4 11
       places de sortie: 7 10
       porte: CALCULATED_DATA.0.1[3]
    transition 10
       places d'entree: 4 11
       places de sortie: 9 10
       porte: CALCULATED_DATA.0.1[3]
    transition 11
       places d'entree: 3 10
       places de sortie: 1 11
       porte: COLLECTED_DATA.0.1[3]
    transition 12
       places d'entree: 2 10
       places de sortie: 1 11
       porte: COLLECTED_DATA.0.1[3]
liste des projections

    projection de l'unite 0
       0 --- <0> ---> ...

    projection de l'unite 1
       1 --- <1> MO_NOTIF.0.0[1] ---> 2
       1 --- <2> OPERAT.0.0[1] ---> 3
       2 ... --- <12> COLLECTED_DATA.0.1[3] --->
1 ...
       3 ... --- <11> COLLECTED_DATA.0.1[3] --->
1 ...

    projection de l'unite 2
       4 ... --- <8> CALCULATED_DATA.0.1[3] --->
5 ...
       4 ... --- <9> CALCULATED_DATA.0.1[3] --->
7 ...
       4 ... --- <10> CALCULATED_DATA.0.1[3] --->
9 ...
       5 --- <7> BASELINE_DATA.0.0[1] ---> 4
       6 --- <6> REQ.0.0[1] ---> 4
       7 --- <5> BASELINE_DATA.0.0[1] ---> 6
       8 --- <4> NOTIF.0.0[1] ---> 4
       9 --- <3> BASELINE_DATA.0.0[1] ---> 8

    projection de l'unite 3
       10 ... --- <11> COLLECTED_DATA.0.1[3] --->
11 ...
       10 ... --- <12> COLLECTED_DATA.0.1[3] --->
11 ...
       11 ... --- <8> CALCULATED_DATA.0.1[3] --->
10 ...
       11 ... --- <9> CALCULATED_DATA.0.1[3] --->
10 ...
       11 ... --- <10> CALCULATED_DATA.0.1[3] ---
> 10 ...

donnees statistiques de l'optimisation

nombre d'unites detruites par l'optimisation E6 :
1 (20 %)
nombre de places detruites par l'optimisation E3
: 7 (37 %)
nombre de transitions detruites par
l'optimisation E3 : 7 (35 %)
```

Uma simulação exaustiva é feita com essa rede de Petri. O grafo de estados (arquivo `.gph`) é apresentado abaixo:

```
graphe pour la specification ``agent''

nombre d'etats: 18
nombre d'arcs: 22

liste des arcs

0 --- <1> MO_NOTIF.0.0[1] ---> 1
0 --- <2> OPERAT.0.0[1] ---> 2
1 --- <11> COLLECTED_DATA.0.1[3] ---> 3
2 --- <9> COLLECTED_DATA.0.1[3] ---> 4
3 --- <13> CALCULATED_DATA.0.1[9] ---> 5
4 --- <13> CALCULATED_DATA.0.1[9] ---> 6
5 --- <3> BASELINE_DATA.0.0[1] ---> 7
6 --- <3> BASELINE_DATA.0.0[1] ---> 8
7 --- <19> i.0.1[0] ---> 9
7 --- <20> i.0.1[0] ---> 10
8 --- <19> i.0.1[0] ---> 11
8 --- <20> i.0.1[0] ---> 12
9 -- <17> CALCULATED_DATA.0.1[9] ---> 13
10 -- <4> NOTIF.0.0[1] ---> 14
11 -- <17> CALCULATED_DATA.0.1[9] --> 15
12 --- <4> NOTIF.0.0[1] ---> 16
13 --- <8> COLLECTED_DATA.0.1[3] ---> 17
14 -- <18> CALCULATED_DATA.0.1[9] --> 13
15 --- <6> COLLECTED_DATA.0.1[3] ---> 17
```

```
16 -- <18> CALCULATED_DATA.0.1[9] --> 15
17 --- <1> MO_NOTIF.0.0[1] ---> 1
17 --- <2> OPERAT.0.0[1] ---> 2

table des etats

marquage = (unite 1, unite 2, unite 3, unite
0)

0 : (-, -, -, 0)
1 : (3, 6, 12, -)
2 : (5, 6, 12, -)
3 : (2, 6, 15, -)
4 : (4, 6, 15, -)
5 : (2, 11, 14, -)
6 : (4, 11, 14, -)
7 : (2, 10, 14, -)
8 : (4, 10, 14, -)
9 : (2, 9, 14, -)
10 : (2, 8, 14, -)
11 : (4, 9, 14, -)
12 : (4, 8, 14, -)
13 : (2, 6, 13, -)
14 : (2, 7, 14, -)
15 : (4, 6, 13, -)
    16 : (4, 7, 14, -)
    17 : (1, 6, 12, -)

statistiques

    nombre d'etats : 18
    taille d'un marquage : 2 octets
    taille d'un contexte : 0 octets
    taille d'un etat : 2 octets
    taille d'un paquet : 5 octets
    nombre de pages : 1
    taille d'une page : 1024 etats, soit 5120 octets
    taille de la table des pages : 16384 entrees, soit
65536 octets
    nombre maximal d'etats : 16777215
    taille de la table de hachage : 8329 entrees, soit
33316 octets
    taille minimale des collisions : 0
       taille maximale des collisions : 1
```

b. Simulação Interativa

A simulação interativa permite a execução de uma especificação através de um único caminho, de tal modo que cada escolha indeterminística é decidida pelo projetista. Um exemplo de tal simulação é apresentado a seguir.

```
/usr/local/lotos/tools/cadp/caesar/com/c
aesar.open agent simulator
caesar.open: using link mode
`simulator' is up to date.
caesar.open: running ``simulator''

<initial state>

*** current state at depth 1

successor #1: MO_NOTIF
successor #2: OPERAT

command 1 ? next

which successor (between 1 and 2) ? 1

*** current state at depth 2

successor #1: i (COLLECTED_DATA [3])

command 2 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 3

successor #1: i (CALCULATED_DATA [9])

command 3 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 4
successor #1: BASELINE_DATA

command 4 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 5

successor #1: i (i)
successor #2: i (i)

command 5 ? next

which successor (between 1 and 2) ? 2

*** current state at depth 6

successor #1: NOTIF

command 6 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 7

successor #1: i (CALCULATED_DATA [9])

command 7 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 8

successor #1: i (COLLECTED_DATA [3])
command 8 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 9

successor #1: MO_NOTIF
successor #2: OPERAT

command 9 ? next

which successor (between 1 and 2) ? 2
```

```
*** current state at depth 10

successor #1: i (COLLECTED_DATA [3])

command 10 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 11

successor #1: i (CALCULATED_DATA [9])

command 11 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 12

successor #1: BASELINE_DATA

command 12 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 13

successor #1: i (i)
successor #2: i (i)

command 13 ? next

which successor (between 1 and 2) ? 2

*** current state at depth 14

successor #1: NOTIF

command 14 ? next

which successor (between 1 and 1) ? 1

*** current state at depth 15

successor #1: i (CALCULATED_DATA [9])

command 15 ? quit
```

6.3.3 Teste do Agente Proativo

O teste pode ser realizado através da execução da especificação. Durante a execução é possível escolher uma dentre três estratégias de teste: 1. O indeterminismo não é permitido; 2. O início do teste é escolhido randomicamente; 3. O início do teste é escolhido pelo projetista. A seguir é apresentado um exemplo onde a estratégia 3 é a escolhida.

```
caesar.open: fetching ``executor.c'' in the source library
rerun ranlib(1)
caesar.open: running ``executor''

maximal number of visible transitions (between 0 and 4294967295) ? 6

available execution strategies:
   (1) deterministic
   (2) non-deterministic with random seed
   (3) non-deterministic with chosen seed
chosen strategy (between 1 and 3) ? 3

chosen seed ? 3

OPERAT                                          MO_NOTIF
BASELINE_DATA                                   BASELINE_DATA
NOTIF                                           OPERAT
<17 states visited, 6 visible transitions executed>
```

6.3.4 Verificação do Agente Proativo

Vários tipos de verificação podem ser realizados para validar a especificação do agente proativo. A seguir é apresentada uma verificação que atesta a ausência de impasse e a equivalência de observação.

Ausência de Impasse

Usando a ferramenta Cæsar.open é possível provar que a especificação do agente proativo não contém impasse.

```
caesar.open: using link mode
/usr/local/lotos/tools/cadp/caesar/bin.sun4/caesar  -open agent
-- caesar 4.9 -- Hubert Garavel (INRIA & IMAG) --
caesar : analyse syntaxique de ``agent''.
.
.
caesar.open: running ``terminator''
order for transition firing (between 0 and 6) ? 6

number of bitmap tables (between 1 and 2) ? 2

size of bitmap 1 (between 0 and 4294967295) ? 66

size of bitmap 2 (between 0 and 4294967295) ? 66

state vector:
marquage = {unite 1, unite 2, unite 3, unite 0}

actual size of bitmap 1: 61

actual size of bitmap 2: 61

*** no deadlock found

bitmap informations at address 0x11768:
size in bits: 61
size in bytes: 8
number of entries with value 0: 60 (98%)
number of entries with value 1: 1 (2%)
number of searches: 3
number of searches with success: 1 (33%)
number of searches with failure: 2 (67%)

bitmap informations at address 0x11790:
size in bits: 61
size in bytes: 8
number of entries with value 0: 60 (98%)
number of entries with value 1: 1 (2%)
number of searches: 3
number of searches with success: 1 (33%)
number of searches with failure: 2 (67%)
```

Equivalência de Observação

A verificação de equivalência é realizada pela operação conjunta da ferramenta Cæsar com a ferramenta Aldébaran. O uso da ferramenta Cæsar permite gerar um autômato no formato adequado para a ferramenta Aldébaran. O uso da ferramenta Aldébaran permite provar que dois autômatos representativos do agente proativo, em diferentes níveis de abstração, são indistinguíveis por um observador externo.

A seguir são apresentados os Sistemas de Transições Rotuladas correspondentes à representação mais abstrata e a uma representação mais detalhada do agente proativo.

- Sistema de Transições Rotuladas da especificação mais abstrata (`proa.aut`).

```
des (0, 10, 7)
(0, OPERAT, 1)
(0, MO_NOTIF, 2)
(1, BASELINE_DATA, 3)
(2, BASELINE_DATA, 4)
(3, i, 0)
(3, i, 5)
(4, i, 0)
(4, i, 6)
(5, NOTIF, 0)
(6,NOTIF,0)
```

Veja a Figura 6.5.

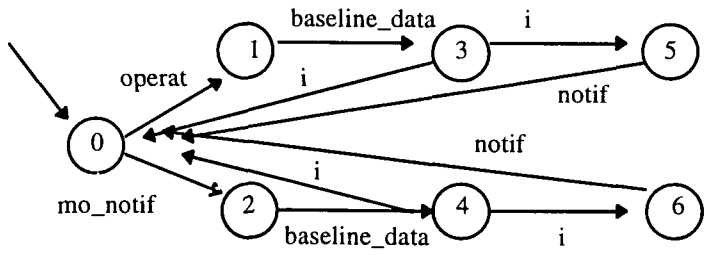


Figura 6.5 - Sistema de Transições Rotuladas da especificação mais abstrata.

- Sistema de Transições Rotuladas da especificação refinada (`proagent.aut`)

```
des (0, 22, 18)
(0, MO_NOTIF, 1)
(0, OPERAT, 2)
(1, i, 3)
(2, i, 4)
(3, i, 5)
(4, i, 6)
(5, BASELINE_DATA, 7)
(6, BASELINE_DATA, 8)
(7, i, 9)
(7, i, 10)
(8, i, 11)
```

```
(8, i, 12)
(9, i, 13)
(10, NOTIF, 14)
(11, i, 15)
(12, NOTIF, 16)
(13, i, 17)
(14, i, 13)
(15, i, 17)
(16, i, 15)
(17, MO_NOTIF, 1)
(17, OPERAT, 2)
```

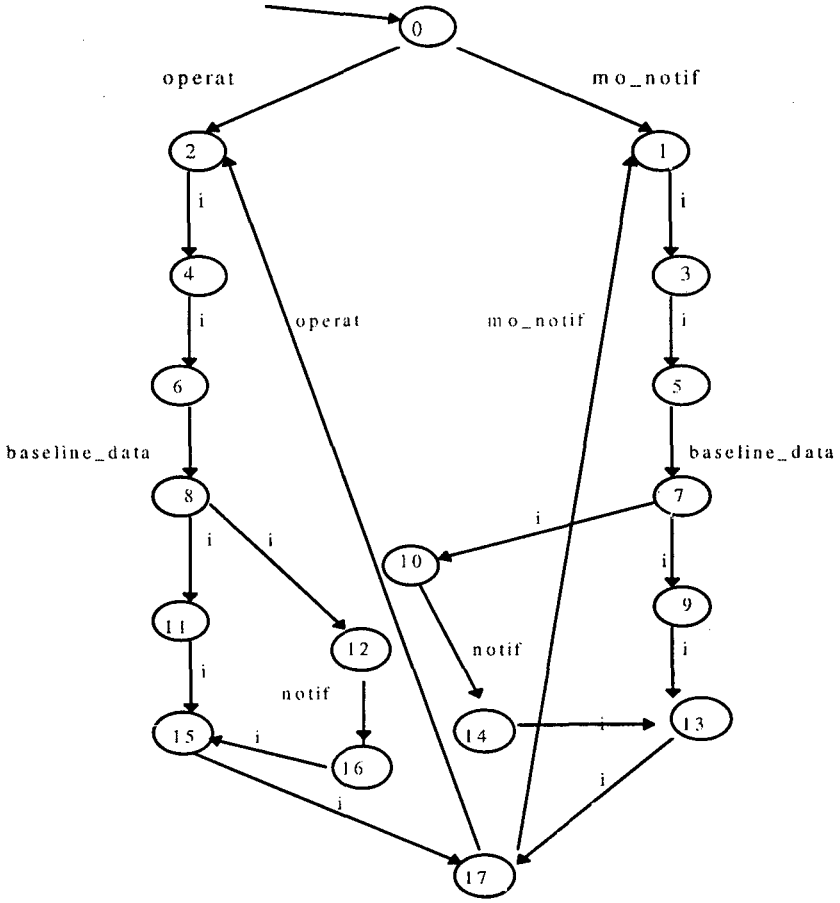


Figura 6.6 - Sistema de Transições Rotuladas da especificação refinada.

Resultado da verificação de equivalência de observação entre as duas especificações (`proa.aut` e `proagent.aut`):

```
aldebaran -oequ proa proagent
TRUE
```

O resultado `TRUE` indica que as duas especificações são equivalentes quanto à observação.

6.3.5 Tradução de LOTOS para código C

Usando a ferramenta TOPO obtém-se automaticamente o código C de referência correspondente a uma especificação LOTOS. O código C que corresponde à especificação LOTOS do agente

proativo tem cento e sete linhas. Algumas dessas linhas de código do arquivo .c são apresentadas a seguir.

```
static char CAESAR_WHAT[]="@(#)CAESAR [open] -- (c) CNRS/IMAG & INRIA -- H. Garavel --
";
#include <stdio.h>
#include <memory.h>
#include <signal.h>
#include <malloc.h>
#define CAESAR_PROJECTION_1(CAESAR_M) (((CAESAR_M)->CAESAR_MARKING_0 & 0xc0) >> 6)
#define CAESAR_PROJECTION_2(CAESAR_M) (((CAESAR_M)->CAESAR_MARKING_0 & 0x38) >> 3)
...
if ((CAESAR_P1->CAESAR_PASS[1]&0x10)) {
CAESAR_S2->CAESAR_MARKING_0=0x44|(0x39&(&(CAESAR_P1->CAESAR_STATE))-
>CAESAR_MARKING_0);
CAESAR_L->CAESAR_TRANSITION_NUMBER = 12;
(*CAESAR_LOOP) (CAESAR_S1, CAESAR_L, CAESAR_S2);
}
}
}
}
#endif
```

7. Conclusões e propostas de novos trabalhos

"Solving the software challenge requires that we treat the many disciplines associated with development as an integrated engineering strategy, not a technical art form" [Evan 89].

A presente dissertação apresenta resultados de uma atividade de pesquisa e desenvolvimento no contexto da engenharia de protocolos com o uso de métodos formais. Nesse contexto é apresentado um método de projeto que propõe a reutilização de construções predefinidas armazenadas em uma biblioteca - a bibLOTOS.

7.1 Conclusões

Uma contribuição efetiva que as instituições de pesquisa podem oferecer para o sucesso e a difusão de LOTOS consiste na disponibilização de ferramentas para microcomputadores compatíveis com o IBM PC. Estas ferramentas devem ser de fácil utilização e integradas ao ambiente de desenvolvimento de sistemas do usuário. Uma vez que tais ferramentas estejam associadas a métodos efetivos de auxílio ao projetista torna-se possível a transferência da tecnologia LOTOS do ambiente acadêmico e de pesquisa para o ambiente industrial.

7.1.1 Análise e interpretação dos resultados alcançados

A análise dos resultados alcançados permite dizer que o desenvolvimento de especificações, com o uso de construções predefinidas, traz inúmeros benefícios, em todas as fases do ciclo de vida de um sistema. Tal se dá porque a trajetória de desenvolvimento de um sistema tem seu tempo encurtado, quando efetuadas operações de reutilização, seja durante a descrição, análise, validação e implementação, seja na manutenção do sistema.

Um dos principais resultados alcançados é a bibLOTOS. As construções predefinidas disponíveis para reutilização, armazenadas na bibLOTOS, já estão validadas, sendo que o projetista deve

garantir somente a correção da especificação final, nos sucessivos refinamentos do sistema em desenvolvimento.

Outro aspecto relacionado aos resultados alcançados diz respeito à experiência adquirida ao escolher como área domínio da aplicação, a gerência de redes. Tal escolha foi muito gratificante, tanto pela aquisição de conhecimentos no estado da arte, quanto pela participação no projeto PLAGERE.

7.1.2 Contribuição Científica

O processo de desenvolvimento de sistemas, a exemplo do que ocorre em relação a outros processos de engenharia, norteia-se pelo paradigma de reutilização. A história das idéias e do progresso intelectual da humanidade fornece um paradigma natural para a contínua reutilização de soluções, já que cada novo avanço repousa necessariamente sobre o conhecimento previamente adquirido através da experiência. O desenvolvimento de especificações, através do uso de construções predefinidas, baseia-se no paradigma da reutilização.

Este processo de desenvolvimento de sistemas abrange métodos, ferramentas e procedimentos. Esta dissertação vem contribuir com este processo através do estabelecimento de um método, e do emprego automático deste método através de uma ferramenta e, além disso, contribui com uma série de procedimentos suplementares. O método define os detalhes de "como fazer" para construir o sistema, a ferramenta bibLOTOS proporciona apoio automatizado ao método e os procedimentos constituem um detalhamento do método onde são apresentadas diversas alternativas de ação.

Ao desenvolver sistemas com o uso de métodos formais, analisando-os rigorosamente e implementando-os automaticamente, o projetista promove a qualidade dos produtos e serviços oferecidos pela indústria.

7.1.3 Adaptação dos usuários à bibLOTOS

O ensino e a difusão de LOTOS em instituições de nível superior, bem como junto à indústria, podem ser estimulados com o uso de ferramentas para PC, integradas em um ambiente de projeto tal como o LOWE.

Os usuários da bibLOTOS devem sentir-se encorajados a ampliar o acervo da biblioteca, embora o acervo atual já permita que diversas especificações LOTOS possam ser obtidas a partir das construções predefinidas já disponíveis na biblioteca. O projetista que queira explorar os recursos da bibLOTOS pode seguir as indicações apresentadas nesta dissertação, sabendo que funções comuns, em diferentes aplicações, podem ser realizadas pelo compartilhamento de um mesmo componente armazenado na bibLOTOS.

7.2 Propostas de novos trabalhos

As redes de computadores e os dispositivos de comunicação constituem ambientes altamente heterogêneos, exigindo mecanismos de gerência padronizados, cada vez mais sofisticados e automatizados. Nesses ambientes ressalta-se a importância de adotar a abordagem proativa nas plataformas de gerência, devido à capacidade de monitorar os recursos e detectar precocemente as circunstâncias que possam afetar as comunicações.

Com a rápida evolução da área de gerência proativa, faz-se necessário o desenvolvimento de algumas novas aplicações, tais como a contabilização de recursos da rede; o controle do congestionamento da interface; a análise no roteamento de pacotes; e o controle de acesso à rede.

O estudo e a especificação formal destas aplicações, bem como de outras voltadas para a gerência de redes de computadores e telecomunicações, é de interesse para a pesquisa.

No presente trabalho desenvolveu-se uma experiência baseada na TDF LOTOS e na ferramenta bibLOTOS. Esses recursos foram utilizados de modo sistemático no projeto de uma aplicação para a gerência proativa de redes. A experiência descrita neste trabalho pode ser aproveitada para a realização de novas aplicações.

Para a criação das construções predefinidas da bibLOTOS foi adotado um processo sistemático, conforme apresentado no Capítulo 5 (Método Proposto para o Desenvolvimento de Sistemas). Este processo é eficiente e permite a definição de outras construções predefinidas, o que levaria a um enriquecimento da bibLOTOS, especialmente se essas novas construções estiverem voltadas para um domínio de aplicação específico como o domínio da gerência de redes. Nesse domínio, por exemplo, poderiam ser definidos vários conjuntos de contruções predefinidas para auxiliar a especificação de objetos gerenciados.

Sugere-se, como continuação das investigações ora iniciadas, que seja realizado um estudo para definir um processo que permita alimentar automaticamente a biblioteca bibLOTOS com novas construções predefinidas. Outro estudo pode ter como alvo definir um método que permita garantir o máximo de eficiência ao uso de construções predefinidas durante a elaboração de projetos.

Para que a reutilização de especificações ocorra é necessário deslocar a ênfase dada à construção individual de sistemas para a construção de componentes reutilizáveis por qualquer aplicação ou sistema. Os componentes "pré-fabricados" podem garantir a alta qualidade do código de programação e reduzir a manutenção decorrente de erros, além de encurtar o ciclo de vida do sistema.

Para padronizar a representação gráfica das construções predefinidas armazenadas na bibLOTOS, pode-se aprofundar os estudos acerca da linguagem gráfica existente para LOTOS, a G-LOTOS.

A migração da bibLOTOS da versão implementada no Microsoft Access 2.0 para o Microsoft Access 7.0 faz-se necessária. Assim a bibLOTOS estará disponível na versão que melhor satisfaça ao usuário, seja ela para Windows 3.1 ou para Windows 95.

A popularização do uso de métodos formais (e de LOTOS em particular) pode ser maior se as ferramentas de auxílio ao emprego desses métodos puderem ser instaladas em microcomputadores. Tendo isso em vista sugere-se que as novas ferramentas sejam desenvolvidas para esse tipo de ambiente.

8. Referências Bibliográficas

"Due to the increase in size, complexity and variety of the software and communication system nowadays, specification methods which can describe the system in a precise, unambiguos and especially abstract (implementation independent) way are becoming more necessary and important in order to design and develop these systems efficiently [TeCh 96]".

[Art 94] **Artola, E.S.** *"Avaliação da Degradação dos Serviços para Realizar uma Gerência Pró-ativa em Redes"*.Trabalho Individual, TI nº 385, CPGCC-UFRGS, fev., 1994.

[Alve 93] **Alves, W. P.:** *"Microsoft Access Interativo"*. Volume 1. São Paulo : Érica, 1993.

[Arno 89] **Arnold, A.** *"Systèmes de transitions finis et sémantique des processus communicants"*. Technique et Sscience Informatiques, Université Bordeaux, 1989, pp. 193-216.

[BaHo 95] **Bauer, M. A.; Katchabaw, M. J.; Hong, J. W.** : *"Behavioural specification and notification enhancements to GDMO"*. Distributed System: Operational and Management, Ottawa, Canadá, 16-18/10/95.

[BoBa 89] **Bochmann, G. v.; Barbeau, M.; Bean, A.; Erradi, M.; Leconte, L.** : *"CRIM/BNR Project---The specification Language MONDEL Progress Report Document N.13 for CRIM/BNR project"*, Abril, 1989.

[BoBr 87] **Bolognesi, T. e Brinksma E.**: *"Introduction to the ISO specification language LOTOS"*. Computer Networks and ISDN Systems, 14(1):25-59, janeiro 1987.

[BoGo 86] **Bochmann, G. v.; Gotzhein, R.** *"Deriving protocol specifications from service specifications"*. Communications, Architectures & Protocols, Proceedings of the ACM SIGCOMM'86 Symposium, Vermont, USA, 1986.

[BoMo 91] **Bochmann, G. v.; Mondain-Monval, P.; Leconte, L.** : *"Formal Description of Network Management Issues"*. Integrated Network Management, II. Elsevier Science Publishers B.V., North-Holland, 1991.

[BoVa 95] **Bolognesi, T.; Van de Lagemaat, J.; Vissers, C.**: *"LOTOSphere: Software Development with LOTOS"*. The Netherlands : Kluwer Academic Publishers., 1995. p.59-85.

[CaMa 94] **Carrilho, J. A.; Madeira, E. R. M.:** *"Um esquema para o gerenciamento do protocolo FTP baseado em domínios"*. XII SBRC, Curitiba PR, 16-20/05/94, pp. 343-362.

[ChCh 94] **Chen, D.J.; Chen D.T.K.**: *"An experimental study of using reusable software design frameworks to achieve software reuse"*, J. Object-Oriented Progr., 7, 1994, pp. 56-57.

[ChCh 96] **Chou, S.; Chen, J.; Chung, C.:** *"A behavior-based classification and retrieval technique for object-oriented specification reuse"*, In: Software-Practice and Experience, vol 26(7), July - 1996. pp. 815-832.

[Coll 89] **Collins, W.** *"OSI management service elements, protocols and application layer structure (ALS)"*. Elsevier Science Publishers, North-Holland, 1989, pp.119-131.

[CoOl 96] **Courtiat, J.-P.; Oliveira, R.C. De**: *"Formal Designs of Multimedia Documents"*. XIV SBRC. 1996.

[DeNo 95] **Delayte, A. P.; Notare, M. S. M. A; Riso B. G.** : "Projeto de Extensões à Plataforma AIX NetView/6000 com a Utilização de Formalismos Algébricos". INFOSUL'95. Porto Alegre, RS, 23-27/10/95.

[DrCh 92] **Drayton, L.; Chetwynd, A.; Blair, G.:** *"Introduction to LOTOS through a worked example"*. School of Engineering Computing and Mathematical Sciences, Lancaster University - UK, 1992.

[EhMa 85] **Ehrig, H., Mahr, B.:** *"Fundamentals of algebraic specifications"*. Springer-Verlag, 1985.

[EiVi 96] **Eijk, P.H.J. van; Vissers, C.A.; Diaz, M.:** *"The formal Description Technique LOTOS"*. Disponível pela Internet via www: <URL:http://www. Elsevier.nl/catalogue/SAC/215/08690/08699/502097/502097.html. Jun, 1996.

[ErFr 91] **Ernberg, P.; Fredlund, L.; Hansson, H.;Jonssson, B.; Orava, F.; Pehrson, B.:** *"Guidelines for specification and verification of communication protocols"*. Swedish Institute of Computer Science, SICS Perspective, report n. 1, 1991, p.34.

[EUCA 96] "EUCALYPTUS 2 - EC-CAN065". Disponível pela Internet via www: <URL: http://www.twente.research.ec.org/esp-syn/txt/ec-can065.html. Jan, 1996.

[Evan 89] **Evans, M.W.** : *"The software factory"*. John Wiley & Sons. New York. 1989. pp. 308.

[FaIm 93] **Fayan, B. L.; Imai, C. T.; Faber, M. B; Lorena, P. S.:** *"Plataforma de suporte a aplicações de gerência"*. Revista Telebrás, 12/1993, pp. 110-121.

[Fe Cu90] **Ferraz, C.A.G. e Cunha, P.R.F.** : *"Um Simulador Funcional para Especificações LOTOS"*. Anais do 8° Simpósio Brasileiro de Redes de Computadores (8°SBRC). Recife, PE. 1990, p. 331-350.

[FeGa 93] **Fernandez, J. *C.*; Garavel, H.; Mounier, L.; Rasse A.; Rodriguez, *C.*; Sifakis, J.:** *"A toolbox for the verification of LOTOS programs"*. LGI-IMAG/VERILOG, França, 1993, p.14.

[Feib 96] **Feibel, W.** : "The Encyclopedia of Networking". The Network Press, 2nd edition. San Francisco, USA. 1996, p.1.315.

[FeMo 90] **Fernandez, J. *C.*; Mounier, L.:** *"Aldébaran: users's manual"*. IMAG-LGI, France, 1989, p.7.

[Fern 89] **Fernandez, J. *C.*:** *"Aldébaran: a tool for verification of communicating processes"*. IMAG-LGI, France, 1989, p.28.

[Fern 90] **Fernandez, J. *C.*:** *"Aldébaran: a tool for verification of communicating"*. IMAG, França, 1990.

[Fran 96] **Franceschi, A.S.M. De** : *"Aplicação de desempenho para validar a Gerência Proativa de Redes"*. Dissertação de Mestrado, CPGCC-UFSC, Florianópolis, 02/1996, p.298..

[FrAu 89] **Freestone, D.; Aujla, S.** : *"Specifying ROSE in LOTOS"*. Integrated Network Management. Elsevier Science Publishers B.V., North-Holland, 1989, PP. 231-245.

[FrIs 93] **Freitas, J. P.; Ishibashi, W. W.:** *"Princípios de construção de* software *para gerência integrada de redes e serviços"*. Revista Telebrás, 12/1993, pp. 103-109.

[FrKo 96] **Franceschi, A.S.M. De, Kormann, L.F., Westphall, C.B.**: *"An Performance Application for Proactive Network Management Architecture."*

In: Proc. of Second IEEE Systems Management Workshop, Toronto, Ontario, Canada, Jun. 19-21, 1996.

[FrKo 96a] **Franceschi, A.S.M De, Kormann, L.F., Westphall, C.B.:** *"Performance Evaluation for Proactive Management Network"*. In: Proc. of the IEEE/ICC'96, International Conference on Communications, Dallas, Texas, USA, Jun. 23-27, 1996.

[FrKo 96b] **Franceschi, A.S.M. de, Kormann, L.F., Westphall, C.B.** *"Performance Evaluation for Proactive Network Management"*. In: Proceedings of the IEEE/ICC'96 International on Communications Conference, Vol. I, Dallas, Texas, Jun., 1996. Pp.

[FrZa 90] **Frachi-Zannettacci, P.;Zarli, A.:** *"An incremental and graphical structure-oriented editor for G-LOTOS."* In: J.Quemada & I.Manas, editors, International Conference on Formal Description Techniques, 3 FORTE'90. North-Holland, 1990.

[FuMa 96] **Furlanetto, E. M.; Maciel, C.; Notare, M. S. M. A.; Riso, B. G.**: *"Projeto Formal de uma Plataforma para Gerência de Redes"*, IV SIC - Seminário de Iniciação Científica, Unijuí, Santa Rosa - RS, 10/10/96, p.203.

[Gara 94a] **Garavel, H.:** *"Cæsar 3.7 reference manual"*. Grenoble - France, 1994, p. 32.

[Gara 94b] **Garavel, H.:** *"CADP manual pages"*. IMAG & INRIA, 18/04/94.

[Gara 94c] **Garavel, H.:** *"The Open Cæsar reference manual"*. Grenoble - France, 04/1994, p.96.

[GaRo 90] **Garavel, H.; Rodriguez, C.:** *"An example of LOTOS specification: the matrix-switch problem"*. LGI-IMAG, Grenoble - França, 1990, p.12.

[Held 92] **Held, G.**: "Network Management - Techniques, Tools and Systems". John Willey & Sons. Chichester, USA. 1992. p.236.

[Henn 88] **Hennessy, H.C.**: *" Algebraic Theory of Processes"*, MIT Press, 1988.

[HiBo 94] **Higashino, T.; Bochmann, G. v.:** *"Automatic analysis and test case derivation for a restricted class of LOTOS expressions with data parameters"*. IEEE Transactions on Software Engineering. Vol.20, No.1, 01/94, pp. 29-42.

[http 95] *"LOTOS NewsLetter"*. Disponível pela Internet via www: <URL: http://wwwtios.cs.utwente.nl/export/lotos/lotos-news/. Abr, 1995.

[http 95] *"OJP - Formal Methods"*. Disponível pela Internet via www: <URL: http://www.vtt.fi/tte/staff/ojp/formal.html. Nov, 1995.

[http 95] *"FME Formal Methods Aplications Database"*. Disponível pela Internet via www: <URL:http://www.cs.ncl-ac.uk/modules/1995-96/csc208/FMEInfRes/ FM-Appl-DB/www/LOTOS.html. Aug, 1995.

[http 95] *"Research Areas"*. Disponível pela Internet via www: <URL: http://www.csi.uottawa.ca/~damyot/phd/areas.html. Sep, 1995.

[http 96] *"LOTOS Information Sources and Projects"*. Disponível pela Internet via www: <URL:htttp://www.cs.helsinki.fi/research/moco/lotos.html. May, 1996.

[http 96] *"LOTOS Static Semantics"*. Disponível pela Internet via www: <URL: http://13awww.romal.infn.it/~mirabell/fdt/HTML/LOTOS_7.html. Fev, 1996.

[http 96] *"Home-Page What is LOTOS?"*. Disponível pela Internet via www: <URL: http://13www.roma1.infn.it/~mirabell/fdt/HTML/simlotos.html. May, 1996.

[http 96] *"ADTs in LOTOS"*. Disponível pela Internet via www: <URL: http://13awww.romal.infn.it/~mirabell/fdt/lotos/papers/adt_fund/adt.fund/node2.html. Fev, 1996.

[ISIE 94] ***ISO* / IEC JTC1 /SC21:** *"Case Resolutions of the ISO/IEC JTC 1; ACSE service definition with extensions to suport the extended application layer structure; ACSE protocol specification with extensions to suport the extended application layer structure; ACSE connectionless protocol specification with extensions to suport the extended application layer structure; Metodology and guidelines for the application layer protocols; Upper layer protocol efficiency"*. Editor John Day, Southampton, United Kingdom, 20-29/07/1994.

[ISO 10040] ***ISO*: "*Information processing systems - Open Systems Interconnection - Systems Management Overview*"**, 1989, p.12.

[ISO 7498.4] **ISO**: *"Information Processing Systems - Open Systems Interconnection - Basic Reference Model"*. 1989. p.185.

[ISO 8649] ***ISO*: "*Information processing systems - Open Systems Interconnection - Service definition for the Association Control Service Element*"**, 1988, p.11.

[ISO 8650] ***ISO*: "*Information processing systems - Open Systems Interconnection - Protocol specification for the Association Control Service Element*"**, 1990, p.26.

[ISO 8807] **ISO**: *"International Standard - IS 8807. Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique*

Based on the Temporal Ordering of Observational Behaviour", 1988.

[ISO 89] **ISO:** *"G-LOTOS: a Graphical Syntax for LOTOS"*. ISO/IEC/JTC1/SC21 N3253. 1989.

[ISO 89] **ISO:** *"G-LOTOS: DAM 1 to ISO 8807 on Graphical Representation for LOTOS"*. ISO/IEC/JTC1/SC21 N4871. jan. 1992.

[ISO 90] **ISO:** *"Proposed Draft Addendum to ISO on G-LOTOS"*. ISO/IEC/JTC1/SC21 N4228. abr. 1989.

[ISO 9072-1] ***ISO:*** *"Information technology - Remote Operations - Part 1: concepts, model and notation"*, 1993, p.46.

[ISO 9072-2] ***ISO:*** *"Information technology - Remote operations - Part 2: OSI realisations - Remote operations service element service definition"*, 1993, p.39.

[ISO 9595] ***ISO:*** *"International Organization for Standardization - Information technology - Open Systems Interconnection - Common management information service definition"*, 1991, p.29.

[ISO 9596] ***ISO:*** *"International Organization for Standardization - Information technology - Open Systems Interconnection - Common management information protocol"*, 1991, p.40.

[Jan 93] **Jander, M.:** *"Proactive LAN Management. Data Communications"*, Mar., 1993.

[KaHi 90] **Kant, C.; Higashino, T.; Bochmann, G. v. :** *"Deriving protocol specifications from service specifications written in Basic LOTOS"*. Montreal,

Canada, 1990, p. 1-16.

[KaHo 95] **Katchebaw, M.J.; Bauer, M.A.; Hong, J.W.:** *"Behavioural specification and notification to GDMO"*, Canada, 1995.

[KaRi 87] **Katz, S; Richter, C.A.; K.-S. The**: *"PARIS: a system for reusing partially interpreted schemas"*, Proc. 9th International Conference on Software Enginneering, 1987, pp. 377-385.

[KhBo 89] **Khendek, F.; Bochmann, G. von; Kant, C.:** *"New results on deriving protocol specifications from services specifications"*. Université de Montréal, 07/1989, p.10.

[KhLi 94] **Khajenoori, S.; Linton, D.G.; Morris, C.A.**: *"Enhancing software reusability through effective use of the essential modelling approach"*, Info. Software Technol., 36, 1994, pp. 495-501.

[Kilo 94] **Kilov, H.:** *"Formal methods and standards"*. Software Engineering Notes, vol. 19 no. 3, 07/94, pp. 40-41.

[KoBo 91] **Kouyzer, A. J.; van den Boogaart, A.K.** : *"The LOTOS framework for OSI Systems Management"*. Integrated Network Management, II. Elsevier Science Publishers B.V., North-Holland, 1991.

[Lang 90] **Langerak, R.:** *"Decomposition of functionality: a correctness preserving LOTOS transformation"*. X International Symposium on Protocol Specification, Testing and Verification, Ottawa - Canadá, IFIP, 1990, pp. 203-218.

[Leon 90] **Léon, G.**: *"On the technology transfer of formal methods: an experience on LOTOS"*. Third International Conference on Formal Description Techniques,

FORTE'90, p.p.567-582, Madrid, 5-8 november, 1990.

[LePe 94] **Lehmann Jr., E. O.; Pedroza, A. de C. P.:** *"Especificação e verificação do protocolo CMIP para gerenciamento de rede"*. XII SBRC, 16-20/05/94, Curitiba PR, pp. 639-658.

[LiLi 96] **Lima, R.M.F.; Lins, R.D.; Queiroz, J.A.M.:** *"Mapping Basic LOTOS Constructors into Occam 2"*, XXIII SEMISCH'96, Recife - PE, 4-9/08/96. pp. 71-82.

[LoCh 92] **Loureiro, A.A.F.; Chanson, S.T.; Vuong, S. T.**: *"FDT tools for protocol development"*. Tutorial. In: Participant's Proceedings of the Fifth International Conference on Formal Description Techniques, FORTE'92, Lannion, França, pp. 38-78, 1992.

[LuBa 89] **Lubars, M.D.**: *"The IDeA design environment"*, Proc. 11th International Conference on Software Enginneering, 1989, pp. 23-32.

[Luce 87] **Lucena, C.:** Inteligência Artificial e Engenharia de Software. Publicações Acadêmico-Científicas - PUC -RJ. IBM Brasil - Jorge Zahar Editor, 1987.

[Mach 93] **Machado, I.:** *"Segurança e gerência de segurança no ambiente TMN"*. Revista Telebrás, 12/1993, pp. 49-55.

[MaMi 93] **Mañas, J.; Miguel, T. de; Robles, T.; Salvachua, J.; Huecas, G.; Veiga, M.:** *"TOPO: Quick reference front-end - version 3R2"*. Universidad. Politécnica de Madrid, 1993, p.31.

[Manã 96] **Mañas, J.A.:** *"A Tutorial on ADT semantics for LOTOS users Part I: Fundamental Concepts"*. Disponível pela Internet via www: <URL: http://13awww.romal.infn.it/~mirabell/fdt/lotos/papers/adt_fund/adt.fund/adt.f

und.html. Mar, 1996.

[Manh 94] **Manhas Jr, E. B.:** *"Uma metodologia para o desenvolvimento de aplicações distribuídas baseadas na técnica de descrição formal ESTELLE"*. Dissertação de Mestrado, EEL-UFSC, 07/94.

[MaNo 96a] **Maciel, C.; Notare, M.S.M.A.; Riso, B. G.**: *"Biblioteca de Construções Predefinidas em LOTOS"*, III JP - Jornada de Pesquisa, Unijuí, Santa Rosa - RS, 10/10/96, p.236.

[MaNo 96b] **Maciel, C.; Notare, M. S. M. A.; Riso, B. G.:** *"Engerharia de Protocolos com o Uso de Construções Predefinidas em LOTOS"*, II CACIC, San Luis, Argentina, 07/11/96, pp. 451-462.

[MaNo 96c] **Maciel, C.; Notare, M. S. M. A.; Riso, B. G.:** *"Biblioteca de Construções Predefinidas em LOTOS - a bibLOTOS"*, III Jornadas de Informática e Investigación Operativa e V Encuentro del Laboratório de Ciência de la Computacion, Montevideo, Uruguay, 11 e 12/12/96.

[Mayr 89] **Mayr, T.:** *"Specification of Object-Oriented Systems in LOTOS"*. Technische Universität Wien - Institut für AngewandleInformatik, Áustria, 1989, pp.107-119.

[MeBo 83] **Merlin, P.; Bochmann, G.v.:** *"On the construction of submodule specifications and communication protocols"*. ACM Trans. on Programming Language and Systems, no 1, 01/83, pp. 1-25.

[MeNo 96] **Mello, A.M.; Notare, M.S.M.A.; Riso, B.G. :** *"Utilização de LOTOS no projeto de Agentes Portáveis para Gerência de Redes"*. I WoSiD, UFBA, Salvador BA, 6-10/04/1996. pp.315-323.

[Meye 88] **Meyer, B.:** *"Object-oriented software construction"*. Prentice-Hall Internacional. 1988.

[Micr 94] **Microsoft Corporation.:** *"Microsoft Access - Guia do Usuário - Versão 2.0"*. 1994. p. 856.

[Miln 80] **Milner, R.** : *"A Calculus of Communicating Systems"*. Volume 92 of Lecture Notes in Computer Science. Springer-Verlag, New York, 1980.

[Miln 89] **Milner, R.:** *"Communication and concurrency"*. Prentice-Hall, New York, 1989.

[MoCl 93] **Moreira, A. M.; Clark, R. G.:** *"Os métodos formais na análise de orientação por objectos"*. Department of Computing Science and Mathematics, University of Stirling - Scothland, 1993.

[MoEb 96] **Moonen, L; Ebrabim, A.:** *"Overview of the specification language LOTOS"*. Disponível pela Internet via www: <URL: http://adam.fwi.uva.nl/ ~leon/lotos_abstract.html. Jun, 1996.

[Neum 96] **Neumann, P.G.:** *"Using formal methods to reduce risks." Communication of the ACM*. V.39, nº 7, 07/96, p.114.

[NoRi 94a] **Notare, M. S. M. A., Riso, B. G.:** *"Utilização de restrições e recursos predefinidos na especificação de protocolos com a TDF LOTOS"*. 12° SBRC, Curitiba PR, 16-20/05/1994, pp. 225-243.

[NoRi 94b] **Notare, M. S. M. A.; Riso, B. G.** *"Transformação de especificações com a técnica de descrição formal LOTOS"*. 2ª Jornada USP - SUCESU-SP de Informática e Telecomunicações, São Paulo SP, 30/05-01/06/94, pp. 263-272.

[NoRi 94c] **Notare, M. S. M. A.; Riso, B. G.:** *"Definição e especificação LOTOS de uma plataforma para dar suporte a aplicações de gerência de redes"*. Telemática'94, Porto Alegre RS, 23-25/08/1994.

[NoRi 94d] **Notare, M. S. M. A.; Riso, B. G.:** *"Uma plataforma para dar apoio a aplicações de gerência em redes heterogêneas"*. XXVII Congresso Nacional de Informática e Telecomunicações. Salvador BA, 21-24/11/94.

[NoRi 95a] **Notare, M. S. M. A.; Riso, B. G.** *"Especificação LOTOS de uma plataforma de apoio à gerência de redes"*. XIII SBRC, Belo Horizonte MG, 22-26/05/95, pp. 483-502.

[NoRi 95b] **Notare, M. S. M. A.; Riso, B. G.:** *"Uma plataforma para integrar aplicações de gerência em redes heterogêneas"*. PANEL'95 - XXI Latin American Conference on Informatics. Canela RS, 29/07 - 04/08/95.

[Nota 95] **Notare, M.S.M.A.**: *"Uma metodologia para a especificação formal de serviços e protocolos de comunicação*". Dissertação de Mestrado, CPGCC-UFSC, Florianópolis, 04/1995, p.298.

[NoTo 95a] **Notare, M. S. M. A.; Tonin, N. A.; Riso, B. G.:** *"Especificação formal de um sistema de gerenciamento de redes heterogêneas"*. IX FENASOFT. São Paulo SP, 17-21/07/95.

[Pech 92] **Pecheur, C.:** *"Using LOTOS for specifying the CHORUS distributed operating system kernel"*. ESPRIT Project, Computer Communications, vol.15, n.2, 03/1992, pp. 93-102.

[PeHo 95] **Perrow, G.S.; Hong, J.W.; Lutfiyya, H.L.; Bauer, M.A.:** *"The abstraction and modelling of management agents*", 1995. pp. 466-477.

[Penn 92] **Penna, M. C.:** *"An overview of IDEA network management platform"*. 11 SBRC, 1992, pp.269-281.

[PeRi 88] **Penedo, M. H.; Riddle, W. E.:** *"Software Engineering Environment Architectures"*. IEEE Transactions on Software Engineering, vol. 14, o. 6, 06/1988, pp. 689-695.

[PhBr 93] **Phifer, L.; Brusil, P.**: *"Incorporating OSI Management Technology into the Marketplace"*. Integrated Network Management, III(C-12), IFIP, 1993.

[PiLo 90] **Pires, L. F.; Lopes de Souza, W.:** *"Step-wise refinement design example using LOTOS"*. Proceedings of FORTE'90, Madrid, 5-8/11/1990, pp.289-306.

[Pire 89] **Pires, L. F.:** *"Projeto de protocolos de inter-rede com o uso da técnica de descrição formal LOTOS - especificações de sistemas intermediários para comunicação com e sem conexão"*. Dissertação de Mestrado, Escola Politécnica da USP - Departamento de Eng. Elétrica, São Paulo SP, pp. 1-260.

[Pire 94] **Pires, L. F.:** *"Architectural notes: a framework for distributed systems development"*. CTIT Ph.D-thesis series No. 94-01. The Netherlands, p. 255.

[PiSi 92] **Pires, L.F.; Sinderen, M. van; Vissers, C.:** *"On the use of pre-defined implementation constructs in distributed systems design"*. University of Twente, 1992.

[PLAG 94] **PLAGERE:** *"Plataformas para gerência de redes"*. Projeto ProTeM-CC, UFSC/CEFET/UFPB/CPqD, 1994.

[Pres 95] **Pressman, S.R.:** Engenharia de Software. MacGraw-Hill. Makron Books do Brasil Editora, 1995.

[QuAz 91] **Quemada, J.; Azcorra, A.; Pavon, S.:** *"Development with LOTOS"*. ESPRIT - *LOTOS*PHERE Project, p. 30.

[QuCu 94] **Queiroz, J.A.M. de; Cunha, P.R.F.**: *"Sistemas Distribuídos: de especificações LOTOS a implementações"*. IX Escola de Computação, 24-31/julho/94. Recife; UFPE-DI, 1994.

[Ramo 94] **Ramos, A. M.:** *"Interface de controle de acesso para o modelo de gerenciamento OSI"*. Dissertação de Mestrado, CPGCC-UFSC, 09/94, p.130.

[ReFr 93a] **Rebelles, P.; Freitas, J. P.:** *"Introdução aos modelos genéricos de arquitetura para rede de gerência de telecomunicações (TMN)"*. Revista Telebrás, 12/1993, pp. 12-23.

[ReFr 93b] **Rebelles, P.; Freitas, J. P.:** *"Modelagem de informação aplicada à gerência integrada de redes de telecomunicações"*. Revista Telebrás, 12/1993, pp. 134-146.

[RiLo 91] **Riso, B.G.; Lopes de Souza, W.:** *"Uma abordagem para a especificação estruturada, em CCS, de sistemas distribuídos"*. Revista Brasileira de Computação, jan/mar 1991, Rio de Janeiro, vol. 6, n. 3, pp. 43-56.

[Riso 91] **Riso, B.G.:** *"Uma abordagem para o design de sistemas distribuídos e protocolos de comunicação"*. Tese de Doutorado, UFPB, Campina Grande PB, 11/91.

[Riso 93] **Riso, B.G.**: *"Definição de processos básicos e sua utilização em especificações LOTOS"*. Monografia. UFSC. 1993, p.28.

[Roch 94] **Rocha, M.A., Westphall, C.B.**: "Gerência de redes de computadores através de novos agentes". In: Anais do 12° SBRC, XII Simpósio Brasileiro de Redes de Computadores, Curitiba, PR, Mai., 1994, p.113-133.

[RoWe 96] **Rocha, M.A., Westphall, C.B** : "Gerência Pró-ativa de Redes de Computadores Utilizando Agentes e Técnicas de Inteligência Artificial" In: Anais do 14° SBRC, XIV Simpósio Brasileiro de Redes de Computadores, Fortaleza, CE, 1996, p.97-117.

[RoMa 96] **Rogerio, K. O.; Maciel, C.; Notare, M. S. M. A.; Riso, B. G.:** "*Especificação LOTOS de objetos gerenciados*". III INFTEL, Petrobrás, Rio de Janeiro RJ, 02-06/12/96. Anais em CD-ROM.

[Rose 91] **ROSE, M. T.:** *"The Simple Book. An Introdution to Management of TCP/IP based Internets"*. USA : Prentice-Hall, 1991.

[SaVi 93] **Saloña, A. A.; Vives, J. Q.; Gómez, S. P.:** *"An introduction to LOTOS"*. Universidad Politécnica de Madrid, 1993, p.37.

[Schm 95] **Schimidt, D.G.:** *"Using design patterns to develop reusable object-oriented communication software". In: Commmunications of the ACM.* Vol.38, october, 1996, pp.65-74.

[ScPi 92] **Schot, J.; Pires, L.F.:** "*Design and implementation strategies*". Memoranda Informática, TIOS, Universiteit Twente, 09/1992, p.35.

[Schw 96] **Schweitzer, C.M.:** "Desenvolvimento de baselines com o uso de simulação para automatização da gerência de redes". Trabalho de Conclusão de Curso, Ciências da Computação, UFSC, Florianópolis, SC, dez, 1996, p. 82.

[SiCh 91] **Sidhu, D.; Chung, A.; Blumer, T.:** "*Experience with formal methods in protocols development*". Computer Communication Review, 1991, pp.81-101.

[SiPi 92] **Sinderen, M.van; Pires, L.F.; Vissers, C.A.:** "*Protocol design and implementation using formal methods*". The Computer Journal, vol.35, n.5, 1992, pp.478-491.

[Sobr 96] **Sobral, J.B.M.:** "*Uma Linguagem para Especificação de Sistemas Distribuídos em Alto Nível de Abstração*", Tese de Doutorado - COPPE/UFRJ, Rio de Janeiro, RJ, 1996, p.233.

[StBo 96] **Steenm, M.W.A.; Bowman, H; Derrick, J.:** "*Composition of LOTOS specifications*". Disponível pela Internet via www: <URL:http:// alethea.ukc.ac.uk/Dept/Computing/Research/NDS/consistency/campos.html. May, 1996.

[Tane 81] **Tanenbaum, A. S.:** "*Computer Networks*". Prentice-Hall, 1981, p. 517.

[Tane 94] **Tanenbaum, A. S.:** "*Redes de Computadores*". Tradução de PubliCare Serviços de Informática. Rio de Janeiro : Campus, 1994.

[TeCh 96] **Termsinsuwan, P.; Cheng, Z.; Shiratori, N.:** "*A next approach to ADT specification support based on reuse of similar ADT by the application of Case-Based Reasoning*". In: Information and Software Technology. Vol.38, 1996, pp.555-568.

[Toni 95] **Tonin, N. A.:** "*Ferramenta TranSP*". FORTE'95. Montreal, Canadá, 17-20/10/95. (poster - demonstração de software).

[Toni 96] **Tonin, N.A.:** "*Engenharia de protocolos com transformação de*

especificações formais: a ferramenta TranSP". Dissertação de Mestrado, Curso de Pós-Graduação em Ciências da Computação-UFSC, 02/1996.

[ToNo 95] **Tonin, N. A.; Notare, M.S.M.A.; Riso, B.G..** "TranSP - Uma ferramenta para transformar especificações de serviço em especificações de protocolo." INFOSUL'95. Porto Alegre, 23-27/10/95.

[ToNo 95a] **Tonin, N. A.; Notare, M. S. M. A.; Riso, B. G.:** *"Transformação de especificações de serviço em especificações de protocolo"*. Trabalho submetido a publicação.

[ToNo 95b] **Tonin, N. A.; Notare, M. S. M. A.; Riso, B. G.:** *"Especificação LOTOS de uma célula flexível de montagem"*. Trabalho submetido a publicação.

[Trei 89] **Treimans, J.:** *"Hippo: a LOTOS simulator"*. University of Twente, The Netherlands, 1989, pp.391-397.

[VaCi 93] **Valenzano, A.; Sisto, R.; Ciminiera, L.:** *"Rapid prototyping of protocols from LOTOS specifications"*. Software - Practice and Experience, vol.23, 01/1993, pp. 31-53.

[Viho 94] **Viho, César Gagnon.:** *"Errors supression in protocols and service preservation"*. SBT/IEEE International Telecommunications Symposium. Rio de Janeiro RJ, 22-26/08/94, pp. 398-402.

[ViSa 93] **Vives, J. Q.; Saloña, A. A.; Gómez, S. P.:** *"Design with LOTOS"*. Universidad Olitécnica de Madrid, 1993, p.37.

[ViSc 88] **Vissers, *C.* A.; Scollo, G.; Sinderen M. v.:** *"Architecture and specification style in formal descriptions of distributed systems"*, University of Twente -

Holanda, 1988.

[ViSc 89] **Vissers, *C.* A.; Scollo, G.; Sinderen M. van; Brinksma, E.:** "*On the use of specification styles in the design of distributed systems*". Proceedings of TAPSOFT'89. Barcelona, 1989.

[Vloe 93] **Vloedt, T. v.d.:** "*The LOTOS toolbox*". AMAST'93. Uviversity of Twente. The Netherlands, 21-25/06/93, pp. 337-338.

[West 94] **Westphall, *C.* B.:** "*Avaliação de plataformas em gerência de redes: OSI, SNMP, SunNet Manager, Openview..*".. SISTEMAS'94, São Paulo - SP, 3-5/05/94, p.24.

[ZiOl 94] **Zirbes, S.F.; Oliveira, J.P.M.:** *"Experimentação em Engenharia de* Software". PGCC-UFRGS. 1994.

[Zirb 94] **Zirbes, S.F.:** *"Reutilização de especificações de sistemas*". PGCC-UFRGS. 1994. pp.111-131.