# Engenharia de protocolos com o uso de construções predefinidas em LOTOS

Cristiano Maciel[1]
crica@inf.ufsc.br

Mirela Sechi Moretti Anonni Notare[2]
mirella@inf.ufsc.br

Bernardo Gonçalves Riso[3]
riso@inf.ufsc.br

Universidade Federal de Santa Catarina (UFSC)
Curso de Pós-Graduação em Ciências da Computação (CPGCC)
Laboratório de Redes e Gerência (LRG)
Cx. P. 476 - CEP.88.040-970 - Florianópolis - SC
Fone: (048) 2319498 Telex: (048) 240 UFSC BR Fax: (048) 2319770

**Resumo:** 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.

**Palavras-chave:** sistemas distribuídos, serviços de comunicação, protocolos de comunicação, gerência de redes, especificação formal, construções predefinidas, ferramentas de projeto, LOTOS, *Access,* bibLOTOS.

**Abstract:** This work presents an approach for designing network management systems by using of bibLOTOS library from pre-defined constructs which are described using the Formal Description Technique LOTOS. The bibLOTOS includes Basic LOTOS constructs (where only behavioural issues are defined) and Full LOTOS constructs (where they are both considered, behavioural and data issues). BibLOTOS is implemented using the *Access* system for microcomputers. An example of bibLOTOS using, in the case of designing a proactive network management system, is presented.

**Keywords:** distributed systems, communication services, communication protocols, network management, formal specification, pre-defined constructs, design tools, LOTOS, *Access,* bibLOTOS, bibLOTOS.

---

1 Aluno do Curso de Pós-Graduação (Mestrado) em Ciências da Computação da UFSC. Bolsista da CAPES.
2 Mestre em Ciências da Computação (UFSC, 1995). Bolsista do CNPq (Projeto PLAGERE).
3 Doutor em Engenharia Elétrica (UFPB, 1991). Professor Adjunto da UFSC.

## 1. Introdução

Sistemas distribuídos de interesse prático são, normalmente, sistemas complexos e de grande porte. O projeto de tais sistemas é, por esse motivo, difícil de realizar. Para que o projeto e a implementação de sistemas distribuídos possam realizar-se de modo rápido e seguro é conveniente a utilização de técnicas de engenharia que empreguem métodos formais [ScSi 92, SiPi 92].
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 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, Leon 90], um padrão internacional da ISO (*International Organization for Standardization*). Para a linguagem LOTOS foram sugeridas diferentes abordagens de utilização. Uma das abordagens considera a possibilidade do uso de construções predefinidas como um recurso de projeto [PiSi 92].
Ao permitir a reutilização de especificações, o uso de construções predefinidas reduz os esforços para a validação de sistemas, proporcionando um desenvolvimento mais veloz [PiSi 92, SiPi 92, ScPi 92, Nota 95]. 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.
Diversos sistemas distribuídos e protocolos de comunicação podem ser 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. A inclusão das construções predefinidas é feita através de um editor (ed mount) disponível no ambiente LOWE.
As construções predefinidas armazenadas na biblioteca bibLOTOS são utilizadas à medida que o projeto do sistema evolui. Desse modo o projetista recolhe as construções predefinidas que são de seu interesse e as integra à descrição do sistema em desenvolvimento.
Este trabalho está organizado do modo seguinte. Na seção 2 apresenta-se um resumo das principais características da TDF LOTOS. Na seção 3 é feita uma avaliação preliminar das principais características da biblioteca bibLOTOS. Na seção 4 propõe-se um conjunto de construções predefinidas. Na seção 5 um exemplo de aplicação de construções predefinidas para o projeto de um sistema de gerência de redes é desenvolvido. Finalmente são apresentadas as conclusões (seção 6) e as referências bibliográficas (seção 7).

## 2. A técnica de descrição formal LOTOS

A linguagem LOTOS [ISO 8807, BoBr 87] é uma das técnicas de descrição formal (TDF) desenvolvidas na ISO por especialistas do ISO/TC97/SC21/WG1/FDT/Subgroup C, tendo sido especificamente concebida para descrever formalmente os serviços e os protocolos do Modelo de Referência OSI (*Open Systems Interconnection*) [ISO 7498.4]. Atualmente considera-se que LOTOS se aplica ao projeto de sistemas distribuídos e concorrentes em geral [FeCu 90].
Em 1988 LOTOS atingiu o estado de padrão internacional [QuCu 94], mas a sua aceitação e utilização em ambientes industriais depende ainda de uma ampla divulgação dos conceitos básicos e avançados da própria linguagem. Nesse ambientes a aceitação de LOTOS pode ser favorecida pela utilização de ferramentas de auxílio à especificação, análise, simulação, verificação, teste e implementação de sistemas. A popularização de LOTOS será maior se essas ferramentas puderem ser instaladas em microcomputadores.
LOTOS é um acrônimo para Language Of Temporal Ordering Specification e apesar do nome talvez sugerir relação com a lógica temporal, a TDF LOTOS se baseia em álgebra de processos [Henn 88]. Devido à sua fundamentação matemática rigorosa, LOTOS, que é uma técnica algébrica de

especificação formal, permite definições claras, concisas, sem ambiguidades e independentes de implementação.
Segundo [QuCu 94], uma especificação LOTOS descreve um sistema por meio de uma hierarquia de definições de processos. Um processo é uma entidade capaz de realizar ações internas e não observáveis, e, além disso, de interagir com outros processos através de ações externamente observáveis que ocorrem em portas de comunicação. A unidade atômica de interação entre processos é denominada evento.
Um evento corresponde a uma comunicação síncrona que pode ocorrer entre processos. Eventos são atômicos no sentido de que eles ocorrem instantaneamente, sem consumir tempo, em um ponto de interação denominado porta, podendo, ou não, envolver troca de valores. A ação não observável externamente é conhecida como ação interna ou evento interno e é representado pela letra **i**.
Em LOTOS sistemas distribuídos são descritos em termos de processos. Neste contexto entende-se que um sistema pode ser representado inicialmente por um único processo, que posteriormente pode ser dividido em vários subprocessos que cooperam entre si, para a realização de algum serviço. Os processos representam a parte dinâmica de um sistema e os tipos de dados representam a parte estática desse sistema.
A especificação da parte dinâmica de um sistema tem a seguinte estrutura em LOTOS:

**specification** NomeEspecificação[portas]:<funcionalidade>
**behaviour**
. . . < expressão-de-comportamento-da-especificação >
**where**
**process** NomeProcesso[portas]:<funcionalidade> :=
. . . < expressão-de-comportamento-de-processo >
**endproc**
...
**endspec**

onde,

| | |
|---|---|
| **specification** | indica o início de uma determinada especificação; |
| NomeEspecificação | é o identificador da especificação; |
| portas | são as portas de comunicação; |
| funcionalidade | é **exit** ou **noexit**, conforme o caso; |
| **behaviour** | indica o início da especificação de comportamento; |
| **process** | indica o início de um processo da especificação; |
| NomeProcesso | é o identificador do processo; |
| **endproc** | indica o fechamento do processo corrente; e |
| **endspec** | indica o fechamento da especificação. |

As expressões e os caracteres não destacados são de responsabilidade do projetista (especificador).

## 2.1. LOTOS Básico

A parte dinâmica da linguagem LOTOS descreve os aspectos de comportamento dos processos e as interações a que estão sujeitos. As expressões de comportamento relatam a ordem em que os eventos devem ocorrer. Conseqüentemente, as regras que devem ser seguidas para a construção dessas expressões constituem a parte essencial de LOTOS Básico. Os processos e os operadores básicos apresentados abaixo fazem parte destas regras.

**Processos Básicos**

| | |
|---|---|
| **stop** | processo completamente inativo; e |
| **exit** | representa um processo que habilita um processo subseqüente. |

**Operadores Básicos**

| | |
|---|---|
| ; | seqüencialidade de eventos; |
| [ ] | escolhas indeterminísticas entre expressões de comportamento; |
| \|\|\| | composição paralela de processos independentes; |
| \|\| | composição paralela de processos dependentes; |

|[ . . . ]| composição paralela geral;
**hide..in** ocultação de portas;
>> habilitação de processo; e
[> interrupção de processo.

### 2.2. LOTOS Completo

A parte estática da linguagem LOTOS descreve os tipos abstratos de dados utilizados na especificação de sistemas, os quais são responsáveis pela representação de valores e estruturas de dados em LOTOS. Tais representações são obtidas com a utilização da linguagem de especificação para tipos abstratos de dados ACT ONE [EhMa 85].
Para a produção de especificações estruturadas, LOTOS dispõe dos seguintes recursos: uma biblioteca de tipos de dados básicos, combinação de especificações, renomeação de especificações, parametrização de especificações, atualização da parametrização das especificações e extensão de especificações com operações e sorts [ISO 8807].
As especificações de tipos de dados algumas vezes são longas e complexas, mas pode-se recorrer à biblioteca de construções predefinidas oferecida no ambiente LOWE, como recurso de projeto.

## 3. Biblioteca bibLOTOS

A concepção de construções predefinidas deve orientar-se para a produção de construções de uso geral. Além disso, deve possibilitar que tanto as construções predefinidas (representadas por especificações LOTOS) quanto as suas implementações correspondentes (em linguagem de programação) sejam armazenadas na biblioteca bibLOTOS.
Fazendo uso desta abordagem, projetistas devem sentir-se encorajados a ampliar a bibLOTOS, sendo que muitas diferentes especificações LOTOS podem ser obtidas a partir das especificações já armazenadas na biblioteca [BoVa 95].

### 3.1. Concepção da bibLOTOS

A bibLOTOS, implementada no ambiente LOWE (LOTOS Windows Environment), armazena construções predefinidas que auxiliam o desenvolvimento de especificações. Para a implementação dessa biblioteca é utilizado o sistema gerenciador de banco de dados relacional para Windows *Microsoft Access*©.
O *Access* é um 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. 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 o *Access* fazer uso intensivo dos recursos gráficos do Windows [Alve 93].
O *Access* utiliza tabelas para armazenar os dados do usuário. Sendo assim, o primeiro passo a ser dado é criar todas as tabelas de dados que farão parte do banco de dados. Em seguida deve-se criar os relacionamentos entre as tabelas e as planilhas de consultas de dados. As telas de entrada de dados, denominadas formulários, podem ser 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.
O Consultor é outra importante característica do *Access*. Enquanto os Assistentes entregam relatórios e formulários já prontos, 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.
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 (nome da extensão de um arquivo Access) [Micr 94]. A escolha do *Access* para implementação da bibLOTOS justifica-se pelo conjunto das facilidades citadas.

### 3.2. Representação da bibLOTOS

Na concepção da bibLOTOS utiliza-se 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, construções usadas no escopo de sistemas de gerência de redes) [NoRi 94a]. Veja a Figura 3.1.

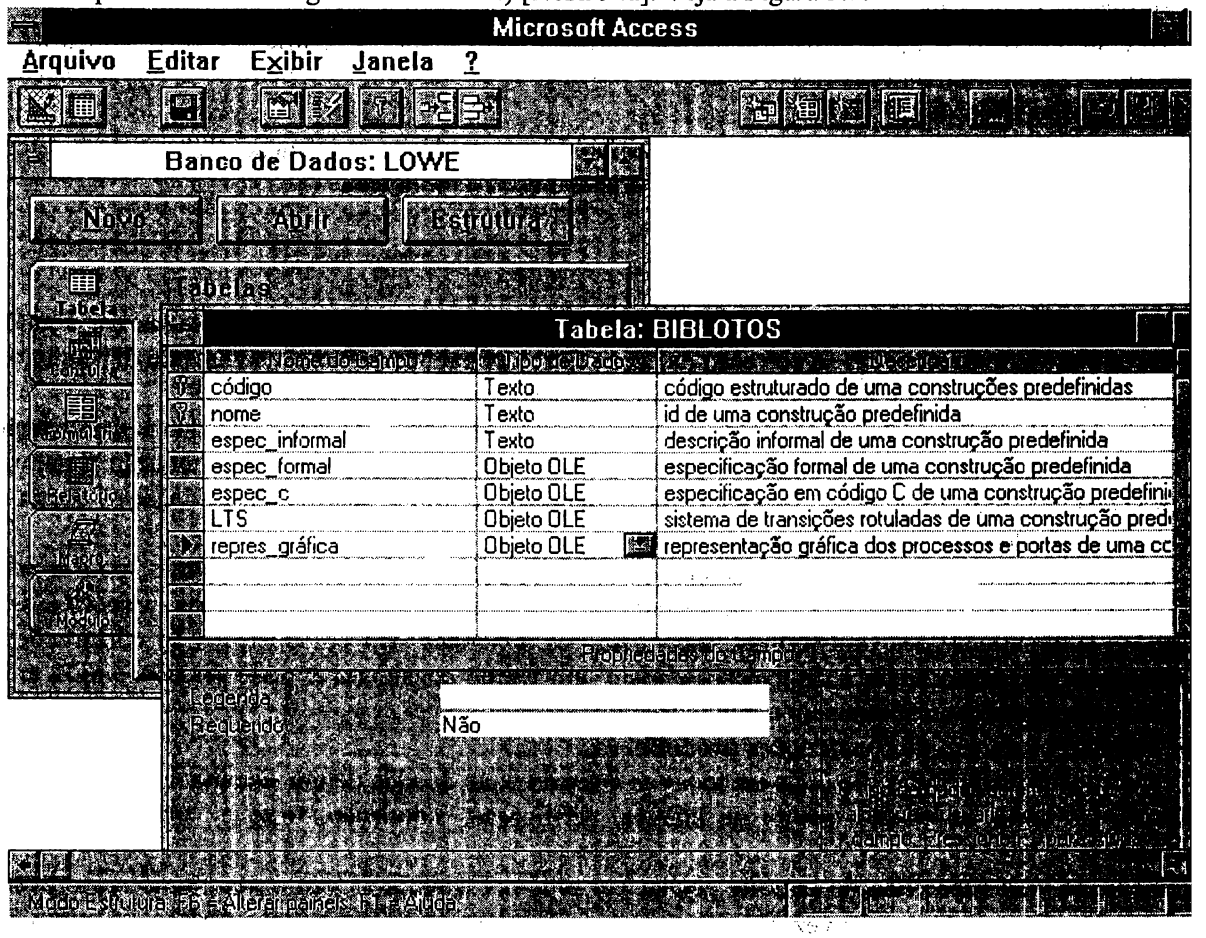


**Figura 3.1.** Gerenciador de Banco de Dados *Microsoft Access*© e bibLOTOS.

Para cada construção predefinida, especificada em LOTOS, e presente na bibliioteca bibLOTOS, tem-se armazenado um conjunto de informações associadas. A idéia é oferecer ao usuário uma ferramenta amigável que possibilite simplificar o processo de construção de uma especificação LOTOS e integrar representações gráficas e textuais. Conforme apresentado na Figura 3.1, a bibLOTOS é implementada com os seguintes campos:

⇨ **Código**: campo de dado tipo texto, o qual armazena o código estruturado da construção predefinida de acordo com a seguinte classificação:

- 1° dígito = LOTOS **B**ásico/LOTOS **C**ompleto;
- 2° dígito = processo **F**inito/processo com possibilidade **I**nfinita;
- 3° dígito = processo **D**eterminístico/**I**ndeterminístico;
- 4° dígito = processo com somente eventos **O**bserváveis/processo com eventos **I**nternos;

Veja exemplo: **process** BFDO_SEQ1[a1] : **noexit** := a1; **stop** **endproc**

Nesse caso o nome do processo indica que se trata de uma construção predefinida em LOTOS Básico (B), com comportamento finito (F), determinístico (D) e sem eventos internos (O).

⇨ **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 das ferramentas.
⇨ **Espec_c**: campo de dado tipo texto, no qual o código em linguagem C correspondente a especificação em LOTOS é armazenada. 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.
⇨ **Repres_gráfica**: campo tipo objeto OLE, no qual representa-se graficamente a construção predefinida. Para esta representação é usada a linguagem G-LOTOS, que permite a representação gráfica de especificações LOTOS.
⇨ **Util**: campo tipo texto, que contém um histórico da utilização da construção predefinida em um sistema e que o especificador vai alimentando durante o uso da biblioteca.

## 4. Construções Predefinidas

A correção das construções predefinidas já está estabelecida antes destas serem usadas em algum projeto específico, de modo que a correção da especificação final torna-se exclusivamente dependente da correção da montagem realizada pelo especificador [BoVa 95].
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). Isto também reduz potenciamente o esforço de validação e encurta o ciclo de vida do projeto de sistemas distribuídos.
A reutilização dos componentes é o ponto alto desta abordagem. 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.

### 4.1. Vantagens do uso de construções predefinidas

As construções predefindas são úteis em todas as fases do ciclo de vida de um sistema, isto é, convém empregá-las tanto na edição quanto na análise, verificação, simulação, teste e manutenção de sistemas.
A **edição** de uma especificação permite ganhar em produtividade, pois não é necessário reescrever uma construção já editada uma vez. Ela é transportada da bibLOTOS para a especificação em curso.
Por exemplo, a fase de **análise** é beneficiada 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 são armazenadas após validadas. Além disso as construções predefinidas são úteis tanto na **simulação** como nos **testes** e **verificações** de sistemas, uma vez que são facilmente inseridas e removidas da especificação, 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 (linguagem C).

### 4.2. Construções predefinidas da bibLOTOS

Os processos abaixo relacionados já fazem parte da biblioteca de construções predefinidas bibLOTOS [Riso 93]. Muitos outros exemplos estão incorporados a esta biblioteca, nos mais diversos estilos de especificação. As construções predefinidas estão sendo agrupadas conforme as suas características principais. Tais caracterísitcas correspondem a um código de classificação.

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

```
(01)  process BFDO_SEQ1[a1] : noexit : = a1; stop  endproc
(02)  process BFDO_FUNCAO_BINARIA[entrada1,entrada2,resultado] : noexit : =
            entrada1;entrada2;resultado;stop      endproc
```

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

```
(03)  process BFIO_ESCOLHA2[a1,a2] :  noexit := a1; stop  [ ]  a2; stop       endproc
(04)  process BFIO_TRI_SEPARADOR[entrada,saida1,saida2,saida3] :  noexit : =
            entrada; (saida1;stop [ ] saida2;stop [ ] saida3;stop) endproc
```

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

```
(05)  process BIDO_CICLICO3[a1,a2,a3] : noexit : =
            a1;a2;a3;BIDO_CLICLICO3[a1,a2,a3]          endproc
(06)  process BIDO_FONTE[liga,sai-valor] : noexit : =
            liga;sai-valor; BIDO_FONTE[liga,sai-valor]   endproc
```

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

```
(07)  process BIIO_DISJUNTOR[a1,a2,a3] : noexit : =
            a1; (a2;BIIO_DISJUNTOR[a1,a2,a3]
             [ ] a3; BIIO_DISJUNTOR[a1,a2,a3] ) endproc
(08)  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
```

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

```
(09)  process BIII_OPCAO_INTERNA[a,b] : noexit : =
            a; BIII_OPCAO_INTERNA[a,b]  [ ]  i;stop  endproc
(10)  process BIII_PREDEFINIDO[notif,baseline_data,mo_notif,operat]
            baseline_data; (i; proa[notif,baseline_data,mo_notif,operat]
                [ ]        i;notif; proa[notif,baseline_data,mo_notif,operat])   endproc
```

## 5. Exemplo de Aplicação

Nesta seção apresenta-se um exemplo que ilustra a utilização de construções predefinidas no projeto de um protocolo de comunicação. Neste exemplo considera-se a especificação de um agente proativo, utilizado no gerenciamento do tráfego de pacotes em redes de comunicação [Tane 94].

### 5.1. Gerência proativa de redes

A gerência de redes deve fornecer mecanismos de monitoração, controle e coordenação de recursos para administrar as atividades realizadas em uma rede, integrando os equipamentos e as informações que ela administra [DeFr 96]. A gerência proativa tem como objetivo prever os possíveis problemas que podem ocorrer na rede e evitar que eles degradem os indícios de serviços oferecidos aos usuários. Portanto, a gerência proativa deve ser capaz de localizar os indícios de problemas antes que os problemas aconteçam, sejam eles de performance, de falhas, de configuração, de contabilização ou de segurança.

A aplicação do gerenciamento proativo se faz através de um sistema complexo, onde muitos componentes específicos são combinados, para realizar esse tipo de gerenciamento. A complexidade da gerência proativa nas redes sugere o uso de rigorosas abordagens de projeto. A engenharia destes sistemas pode ser feita com vantagens através do uso de Técnicas de Descrição Formal (TDF's) [BoMo 91, LoCh 92].

Um agente proativo é um dos componentes de uma aplicação de gerência proativa [Art 94]. Uma aplicação desse tipo pode envolver, além do agente proativo, outros componentes tais como uma MIB (*Management Information Base),* uma *BASELINE* (que tem uma representação do comportamento normal da rede) e uma Plataforma de Gerência.

Este trabalho desenvolve o estudo de uma aplicação para o gerenciamento de redes, através da especificação formal, com o uso de construções predefinidas, dos serviços de gerência proativa de redes.A aplicação especificada permite o monitoramento do tráfego de mensagens de pacotes na rede adotando a abordagem proativa. A especificação formal desta aplicação faz uso das construções predefinidas presentes na bibLOTOS.

### 5.1.1. Descrição do agente

Em alto nível de abstração o agente proativo é visto como um sistema onde somente as portas de entrada e saída de dados são observáveis. Por não se observar a estrutura interna do sistema é que esta estrutura pode ser denominada de caixa preta. Veja a Figura 5.1.

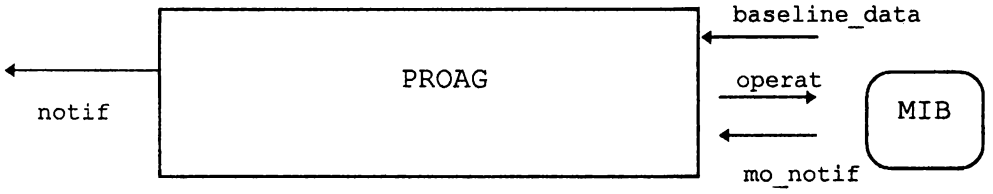


**Figura 5.1.** Representação gráfica do agente proativo em alto nível de abstração.

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 e compara os valores obtidos com os valores armazenados na *BASELINE* (o acesso aos valores armazenados é representado através de eventos na porta *baseline_data*) [DeKo 96a].

As notificações (representadas por eventos na porta *notif* ) são enviadas para a Plataforma de Gerência. Eventualmente, a MIB pode enviar notificações para o agente proativo PROAG (neste caso ocorre um evento na porta *mo_notif* ), quando, por exemplo, o valor de um parâmetro excede um limite preestabelecido.

Considerando o agente proativo PROAG da Figura 5.1, a especificação formal correspondente em LOTOS pode ser apresentada como segue. Nesse caso o PROAG é especificado em alto nível de abstração.

```
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 (*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 *GET* (através da porta *mo_notif*), quando necessário.
Após esta escolha o agente obtém informações sobre a BASELINE através de um evento na porta *baseline_data*. Ocorre então uma 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 a Plataforma de Gerência antes que o processo *proa* seja chamado recursivamente.

```
process proa[notif,baseline_data,mo_notif,operat] : noexit :=
      operat; baseline_data;                          (*realiza operações sobre os objetos
                                                       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; baseline_data;                       (*recebe informações da MIB 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*)
endproc
```

A especificação do agente proativo PROAG pode ser dividida em duas partes. Cada uma dessas partes pode ser especificada com a utilização de uma instância de uma construção predefinida (BIII_PREDEFINIDO) armazenada na bibLOTOS.

```
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]
                  baseline_data;
                  (i; proa[notif,baseline_data,mo_notif,operat]
                  []
                   i;notif; proa[notif,baseline_data,mo_notif,operat])
            endproc
    endproc
endspec
```

**5.1.2. Descrição detalhada do agente**

Para refinar o *proactive_agent* pode-se considerar 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_data* e os valores

coletados, constantemente, pelo *remote_monitor*. As notificações (*notif* ) são enviadas para a Plataforma de Gerência através de eventos na porta *notif*. Veja a Figura 5.2.

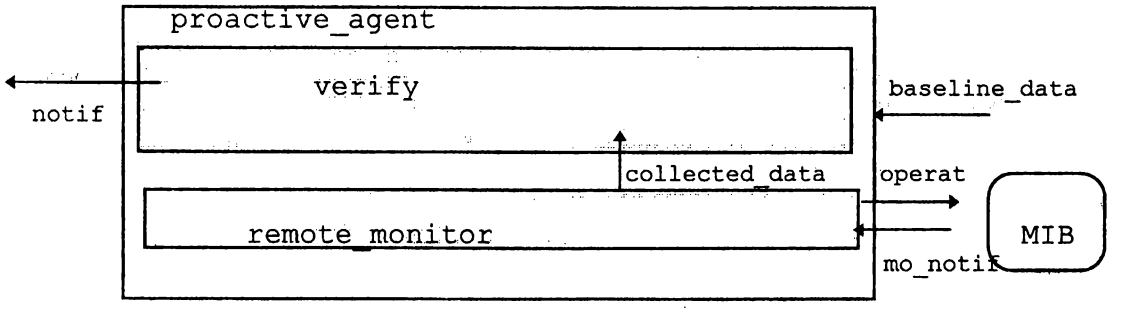


**Figura 5.2.** Representação gráfica do agente proativo.

Ao refinar o componente *verify* pode-se observar a existência dos processos *compare* e *calculate*, como apresentado na Figura 5.3.

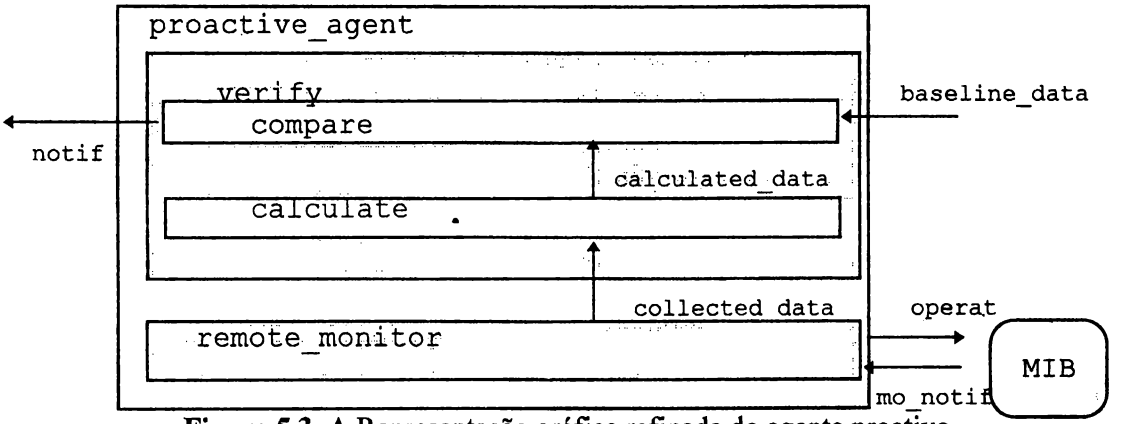


**Figura 5.3.** A Representação gráfica refinada do agente proativo .

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_dat,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
```

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* dois processos em composição geral (operador |[ ]| ): *remote_monitor* e *verify*, comunicam-se através da porta oculta *collected_data* (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 *GET*) ao *remote_monitor*, representada por um evento na porta *mo_notif*.

O processo *verify* é composto de dois processos: *calculate* e *compare*. 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 um evento interno **i** ou é enviada uma notificação para o gerente neste caso ocorre um evento na porta *notif*), ou os dados são calculados e comparados novamente, para então chamar o processo *compare* recursivamente.

Os processos compare, calculate e remote_monitor são instâncias de construções predefinidas (processos predefinidos) armazenados na bibLOTOS.

## 6. Conclusões

Este trabalho contribui com atividades de pesquisa e desenvolvimento para a área de engenharia de protocolos. Nesse contexto é proposta uma metodologia de projeto baseada em construções predefinidas, através da utilização da Técnica de Descrição Formal LOTOS.

Uma biblioteca de construções predefinidas (biblioteca bibLOTOS) busca favorecer o estabelecimento de uma metodologia de projeto e desenvolvimento de sistemas distribuídos que emprega a TDF LOTOS. Essa metodologia é de uso geral mas, neste trabalho, ela é ilustrada através do projeto de um sistema de gerência proativa para redes de comunicação.

Pretende-se, na continuação das pesquisas realizar um estudo sobre métodos automáticos para alimentar a bibliteca com construções predefinidas. Outro estudo pretende definir um método que permita automatizar o uso construções predefinidas durante a elaboração de projetos.

Uma contribuição efetiva para o sucesso e a difusão de LOTOS, consiste na disponibilização de ferramentas para PC, amigáveis e integradas ao ambiente de desenvolvimento de especificações LOTOS. Uma vez desenvolvidas metodologias efetivas e ferramentas adequadas de auxílio ao especificador será possível a transferência da tecnologia LOTOS do ambiente acadêmico para o ambiente industrial.

## 7. Referências Bibliográficas


[Alve 93] **Alves, W. P.:** *"Microsoft Access Interativo"*. V. 1. São Paulo : Érica, 993.

[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.

[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.

[DeFr 96] **De Franceschi, A.S.M.**: *"Aplicação de desempenho para validar a Gerência Proativa de Redes"*. Trabalho individual para encaminhamento de tese de mestrado. Florianópolis, UFSC-CPGCC. Março 1995.

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

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

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

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

[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.

[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.

[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.

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

[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.

[Nota 95] **Notare, M.S.M.A.** : *"Uma metodologia para a especificação formal de serviços e protocolos de comunicação"*, Dissertação de Mestrado, Curso de Pós-Graduação em Ciências da Computação-UFSC, 04/1995, p.298.

[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.

[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.

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

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

[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.

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