Universidade Federal de Santa Catarina

Programa de Pós-Graduação em Ciência da Computação

Novas Funções de Gerência para Redes de Telecomunicações e de Alta Velocidade usando a Plataforma OSIMIS

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

Luiz Fernando Kormann

Orientador
Prof. Dr. Carlos Becker Westphall

Florianópolis, março de 1997

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, na área de concentração Sistemas de Computação e aprovada em sua formal final pelo programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Novas Funções de Gerência para Redes de Telecomunicações e de Alta Velocidade usando a Plataforma OSIMIS

Prof. Dr. Carlos Becker Westphall
Orientador

Prof. Dr. Murilo Silva de Camargo
Coordenador do Curso

Comissão Examinadora:

Prof. Dr. Carlos Becker Westphall
(Presidente)

Prof. Dr. Bernardo Gonçalves Riso

Prof. Dr. João Bosco Mangueira Sobral

Prof. Dr. Joberto Sérgio Barbosa Martins

"Tous le jours on va
de mieux en mieux à
tous les points de vue"

Este trabalho é dedicado a você:
Carlos Becker Westphall.

Agradecimentos

Gostaria de agradecer aos professores Bernardo Gonçalves Riso, Jõao Bosco Mangueira Sobral e Joberto Sérgio Barbosa Martins pelo interesse e pelas discussões sobre partes deste trabalho.

A construção dos serviços propostos neste trabalho exigiu um considerável esforço de implementação em linguagens C e C++. Eu gostaria de agradecer ao Alvaro Sampaio Corrêa Neto, Ariadne M. Raschcowetzki, Clarissa Carneiro Mussi, Juliano Fontoura Pereira, Ketter Ohnes Rogério, Nelson Sinibaldi Filho, Viviane Noemi Mazzorca e Robinson Mittmann pela importante contribuição na construção dos serviços desenvolvidos neste trabalho.

Gostaria de agradecer também aos membros de Departamento de Ciência da Computação da *University College London,* Sr. George Pavlou e Sr. David Griffin pelos seus inúmeros ensinamentos.

Em especial, agradeço a orientação do professor Carlos Becker Westphall.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ACSE - Association Control Service Element

API - Application Program Interface

ASN.1 - Abstract Syntax Notation. One

ATM - Asynchronous Transfer Mode

BRISA - Sociedade Brasileira para a Interconexão de Sistemas Abertos

CMIP - Common Management Information Protocol

CMIS - Common Management Information Service

CMISE - Common Management Information Service Element

CMOT - CMIP over TCP/IP

COM - Common Object Model

CORBA - Common Object Request Broker Architecture

DCE - Distributed Communication Equipment

DCN - Data Communication Network

DME - Distributed Management Environment

DSA - Directory Service Agent

DSAP - Directory Service Access Point

DSS - Directory Service Support

DUA - Directory User Agent

GDMO - Guidelines for the Definition of Managed Objects

GMS - Generic Managed System

HP - Hewlett Packard

IAB - Internet Activities Board

IBM - International Business Machine

IDL - Interface Description Language

IEC - International Electrotechnical Commission

IP - Internet Protocol

IS - Intermediate System

ISDN - Integrated Services Digital Network

ISM - Integrated System Management

ISO - International Organization for Standardization

ISODE - ISO Development Environment

ITU - International Telecommunications Union

KS - Knowledge-Source

LME - Layer Management Entity

MD - Mediation Device

MDB - Management Data Base

MIB - Management Information Base

MIM - Management Information Model

MIT - Management Information Tree

MSAP - Management Service Access Point

MO - Managed Object

NE - Network Element

NEMESYS - Network Management Using Expert Systems

NMF - Network Management Forum

NNI - Network-to-Node Interface

OID - Object Identifier

OAM - Operation, Administration and Maintenance

OMA - Object Management Architecture

OMG - Object Management Group

OS - Operation System

OSF - Open Software Foundation

OSI - Open Systems Interconnection

OSIMIS - OSI Management Information Service

PC - Personal Computer

PROFF - Primary Rate ISDN OSI Office Facilities

QA - Q-Adaptor

RDN - Relative Distinguished Name

RFC - Request For Comments

RMIB - Remote MIB

ROSE - Remote Operations Service Element

RPC - Remote Procedure Call

SDH - Synchronous Digital Hierarchy

SGBD - Sistema Gerenciador de Base de Dados

SMA - System Management Agent

SMAE - System Management Application Entity

SMASE - System Management Application Service Element

SMI - Structure of Management Information

SMIB - Shadow MIB

SML - System Management Language

SMO - Shadow Managed Object

SNMP - Simple Network Management Protocol

VC - Virtual Channel

VCC - Virtual Channel Connection

VCI - Virtual Channel Identifier

VP - Virtual Path

VPC - Virtual Path Connection

VPI - Virtual Path Identifier

TCP - Transmission Control Protocol

TMN - Telecommunications Management Network

XOM - X/Open Object Management

UDP - User Datagram Protocol

UNI - User-to-Network Interface

Resumo

Com o objetivo de explorar as interfaces para a construção de processos de gerenciamento existentes na plataforma OSIMIS e demais ferramentas para a construção de aplicações de gerência foram desenvolvidas extensões à plataforma. A implementação destes novos serviços envolveu o emprego de parte dos componentes fornecidos pelo OSIMIS e forneceu subsídios para o posterior desenvolvimento de aplicações de gerência. Em acréscimo, foi também proposto um método para o desenvolvimento de aplicações OSI, mais especificamente, processos que utilizam interfaces de alto nível para o acesso ao protocolo CMIP. O método proposto engloba os aspectos sintáticos e semânticos para a construção dos objetos gerenciados e aplicações de gerenciamento. Estão incluídas nas etapas de desenvolvimento as interfaces para a construção de processos gerentes e agentes e os compiladores ASN.1 e GDMO. A abordagem inicial adotada neste trabalho para o desenvolvimento de aplicações para redes de telecomunicações serviu para a implementação de serviços de contabilização. Estes serviços estão em conformidade com o modelo de gerenciamento OSI e seguem os requisitos da Função de Medida de Uso [ISO 10164-10b]. O objetivo principal destes serviços de contabilização resume-se no fornecimento de uma estrutura de suporte ao desenvolvimento de aplicações de gerência. Esta extensão de contabilização desenvolvida para a plataforma OSIMIS fornece toda a atividade de controle de coleta de informações sobre a utilização de recursos. Desta forma, uma vez configurado o cenário de contabilização fica sob responsabilidade do projetista da aplicação somente a representação, por meio de objetos gerenciados, dos recursos a serem gerenciados. As atividades realizadas em gerência de contabilização forneceram subsídios para o desenvolvimento de funções complexas que envolvam os recursos da arquitetura OSI e o emprego de interfaces e ferramentas para a construção de processos de gerenciamento. Uma vez conhecidos os procedimentos para o desenvolvimento de serviços TMN de acordo com os requisitos apresentados pela plataforma OSIMIS, esforços foram concentrados com o objetivo de implementar funções de configuração para redes de alta velocidade (ATM).

Abstract

With the objective to explore the interfaces and the supporting tools for the construction of management applications present in the OSIMIS platform, extensions to the platform were developed. The implementation of these new services employed the whole set of facilities contained in the OSIMIS platform and provided support for the further development of management applications. In addition, a method for the development of OSI applications was also proposed, more specifically processes that make use of high level interfaces for the CMIP protocol. This method encompasses both syntactic and semantical aspects for the construction of managed objects and management applications. The development phase of this method includes the interfaces for the construction of manager and agents as well as the ASN.1 and GDMO compilers. The initial approach adopted in this work for the development of applications for telecommunication networks involved the construction of accounting services. Such services are in conformity with the OSI management model and follow the requirements present in the Usage Metering Function [ISO 10164-10b]. The main benefit obtained from these accounting services consist on the provision of a supporting structure for the development of accounting applications. These new accounting services developed for the OSIMIS platform also provide the capability to control and collect the information over the usage of a resource. Therefore, once configured the accounting scenario, the developer of an accounting application is then only responsible for the representation, through managed objects, of the resources to be managed. The activities on accounting management provided knowledge for the development of complex applications involving the concepts of the OSI architecture and the employment of interfaces and tools for the construction of management processes. Moreover, once achieved the proceedings for the development of TMN services according to the requirements of the OSIMIS platform, efforts had been driven to the construction of configuration functions for high speed networks, as in the case of the ATM architecture.

1. Introdução

Com o desenvolvimento da microeletrônica, dos circuitos integrados e das tecnologias de transmissão de dados, tornou-se possível a distribuição do processamento. Os dispendiosos computadores de grande porte, que representavam os únicos recursos computacionais da década de 60, cederam lugar aos mini e microcomputadores, que associados às técnicas de comunicação de dados, tornaram as redes de computadores uma realidade.

Muito já se desenvolveu em termos de *software* e *hardware* desde que as primeiras redes foram projetadas e implantadas. No entanto, um dos problemas emergentes consiste na heterogeneidade e falta de interoperabilidade dos recursos computacionais envolvidos (*hardware*, *software* e dados). Em geral, equipamentos de fabricantes diferentes possuem características distintas, utilizam *softwares* específicos e manipulam dados com formatos incompatíveis. Até mesmo equipamentos idênticos de um só fabricante, quando empregados em aplicações distintas, podem apresentar características heterogêneas.

Cada fabricante propôs uma arquitetura de rede proprietária, tentando estabelecê-la como padrão no mercado. Os usuários, após implantarem determinado sistema, ficavam restritos a um fabricante, sem condições de expandir seu sistema utilizando outras arquiteturas.

Com o objetivo de solucionar problemas de interoperabilidade entre sistemas de gerência de rede, surgiram diversas propostas, dentre as quais destacam-se os modelos *Internet* e OSI (*Open Systems Interconnection*). O primeiro, proposto pela IAB (*Internet Activities Board*), é de simples implementação e

tornou-se um padrão *de facto* no mercado. O segundo, definido pela ISO (*International Organization for Standardization*), é o modelo mais abrangente e apresenta extensa documentação por meio de normas e recomendações que descrevem os requisitos de informação, físicos e funcionais do serviço a ser implementado.

Além de garantir a interoperabilidade entre os sistemas heterogêneos, é necessário também prover meios para monitoração e controle dos recursos envolvidos. Na maioria dos ambientes, o gerente de rede supervisiona o funcionamento de cada recurso através de *softwares* dedicados. Sob a responsabilidade do gerente é mantida a análise das informações fornecidas por estes *softwares.* Estas informações representam os únicos dados disponíveis ao gerente para que este possa realizar ações que objetivam manter a rede em funcionamento.

Existem atualmente no mercado, diversos sistemas proprietários destinados ao gerenciamento de redes, tais como: *Net View* (IBM), *SunNet Manager* (Sun *Microsystems*) e *OpenView* (HP). Estes sistemas apresentam também a restrição de não serem interoperáveis, e geralmente não operarem satisfatoriamente em máquinas de outros fabricantes.

A falta de interoperabilidade no gerenciamento de recursos de uma rede reflete nos altos custos de manutenção, diminui a eficiência na monitoração dos equipamentos utilizados e impossibilita a implantação de medidas de segurança mais eficazes. O conceito de Gerenciamento Integrado de Redes, surge com o objetivo de prover mecanismos que possibilitem a monitoração e o controle dos vários recursos da rede, independente das diferenças entre fornecedores e serviços

suportados. A gerência deixa de ser feita de maneira isolada, passando a ser suportada por uma Plataforma de Gerenciamento.

Esta plataforma deve ser capaz de fornecer informações em tempo real, emitir relatórios, além de realizar ações corretivas para recuperar falhas na rede. O funcionamento da plataforma não deve causar impactos na performance, que venham a degradar o desempenho dos serviços da rede.

Como exemplos de soluções que provêm gerenciamento integrado, encontramos as ferramentas ISODE (*ISO Development Environment*) e OSIMIS (*OSI Management Information Service*). A primeira fornece uma plataforma para o desenvolvimento de aplicações OSI sobre a arquitetura TCP/IP (*Transfer Control Protocol/Internet Protocol*). A segunda, construída sobre o ISODE, apresenta um conjunto de classes e funções que fornece suporte ao desenvolvimento de aplicações de gerenciamento OSI, seguindo o modelo gerente-agente.

O modelo de gerenciamento OSI, não está limitado ao tratamento de falhas, apresentando um modelo funcional que inclui também aspectos de desempenho, configuração, segurança e contabilização.

Com o objetivo de explorar as interfaces para a construção de processos de gerenciamento existentes na plataforma OSIMIS e demais ferramentas para a construção de aplicações de gerência (i.e. compiladores ASN.1 e GDMO) foram desenvolvidas extensões a plataforma. A implementação destes novos serviços envolveu o emprego de parte dos componentes fornecidos pelo OSIMIS e forneceu subsídios para o posterior desenvolvimento de aplicações de gerenciamento. Em acréscimo, foi também proposto um método para o desenvolvimento de aplicações

OSI, mais especificamente, processos que utilizam interfaces de alto nível para o acesso ao protocolo CMIP.

Uma prática muito adotada em empresas de telecomunicações consiste no estabelecimento de tarifas para o fornecimento de serviços. Em algumas redes de computadores o fornecimento de recursos é monitorada não somente para o estabelecimento de tarifas, como também para o controle e a distribuição de cotas de consumo aos usuários. Para a realização destas atividades deve existir um mecanismo de contabilização que seja genérico o suficiente, de forma a possibilitar o processamento dos dados de acordo com os requisitos dos sistemas onde a atividade de contabilização seja necessária. Desta forma, o mecanismo deve ser capaz de contabilizar o uso de equipamentos e recursos característicos tanto de uma rede local como de uma rede de telecomunicações.

A primeira abordagem no desenvolvimento de extensões a plataforma OSIMIS envolveu a implementação de serviços de contabilização para ambientes de telecomunicações como proposto pela recomendação ISO/IEC 10164-10. Uma vez conhecido todo o procedimento para o desenvolvimento de serviços TMN de acordo com os requisitos apresentados pela plataforma OSIMIS, a atividade subseqüente consistiu no estudo da viabilidade da construção de serviços TMN em ambientes de redes de alta velocidade. Esforços foram concentrados com o objetivo de implementar funções de configuração para o estabelecimento de canais e caminhos virtuais em redes ATM.

A utilização de conceitos TMN para a gerência de redes de alta velocidade está em conformidade com pesquisas recentes [AyTa 96] [Ku 96] [FuYo 96] que analisam o emprego de técnicas adequadas para o gerenciamento do futuro ambiente de redes ATM. As interfaces fornecidas pelo protocolo CMIP associadas

com a característica dos componentes distribuídos da arquitetura TMN contribuem para a construção de aplicações complexas que possam sustentar o alto desempenho de uma rede ATM.

Esta dissertação de mestrado está estruturada em nove capítulos. Após a introdução (primeiro capítulo) seguem os demais capítulos apresentados a seguir.

O Segundo Capítulo apresenta uma análise sobre os componentes das plataformas de gerenciamento de redes comumente adotadas na monitoração e controle de redes. Os critérios escolhidos para a avaliação destas plataformas caracterizam os aspectos relativos às aplicações implementadas, aos serviços oferecidos, às ferramentas de desenvolvimento e implementação e demais requisitos de portabilidade e interoperabilidade.

O Terceiro Capítulo fornece uma descrição da plataforma de gerenciamento OSIMIS, onde são discriminadas as aplicações de gerenciamento disponíveis na plataforma e as interfaces que fornecem suporte ao desenvolvimento de processos gerentes e agentes.

O Quarto Capítulo propõe um método para o desenvolvimento de extensões à plataforma OSIMIS. Neste capítulo são descritas todas as etapas envolvidas no desenvolvimento de objetos gerenciados e os detalhes de integração destes com os processos de gerenciamento.

O Quinto Capítulo apresenta detalhadamente a Função de Medida de Uso e, por seguinte, os métodos utilizados para implementação de serviços de contabilização para a plataforma OSIMIS.

O Sexto Capítulo descreve de forma sucinta os principais componentes do modelo TMN para a gerência de redes de telecomunicações. Os aspectos

abordados neste capítulo fornecem subsídios para a compreensão dos conceitos TMN empregados no gerenciamento de configuração de redes de alta velocidade.

O Sétimo Capítulo descreve os conceitos definidos pelo TMN para a gerência de configuração de Canais Virtuais e Caminhos Virtuais em um ambiente de redes ATM.

O Oitavo Capítulo apresenta os aspectos de implementação de um Elemento de Rede TMN.

O Nono Capítulo apresenta as conclusões e propostas de trabalhos futuros, e por fim, são incluídas as referências bibliográficas e os anexos.

2. Plataformas de Gerenciamento

Fazem parte do universo de padronização os modelos de gerenciamento definidos na arquitetura OSI (*Open Systems Interconnection*) proposto pela ISO (*International Organization for Standardization*) e o modelo de gerência *Internet* proposto pela IAB (*Internet Activities Board*). Em acréscimo, também fazem parte deste universo de padronização, as recomendações que sugerem um ambiente multi-protocolar para a gerência de redes tais como o modelo de informação DME (*Distributed Management Environment*) definido pelo OSF (*Open Software Foundation*), e demais proposições do NMF (*Network Management Forum*) e *OMNIPoint*.

Tais proposições apresentam diferenciais que as caracterizam para a atuação em ambientes de gerenciamento distintos. Como exemplo, conceitos do modelo de gerenciamento OSI são empregados para a gerência de redes de telecomunicações TMN (*Telecommunication Management Network*), enquanto que o modelo *Internet* está estabelecido no mercado como padrão *de facto* e amplamente utilizado em redes de computadores.

Em paralelo, as principais empresas provedoras de equipamentos computacionais lançam no mercado sistemas proprierários para o suporte e desenvolvimento de aplicações de gerenciamento. Embora, tais plataformas apresentem elementos comuns que sugerem a interoperalidade com sistemas de diferentes fornecedores, estas, até então, não disponibilizam os requisitos de operabilidade, interoperabilidade e portabilidade que garantem um sistema aberto para a gerência do atual ambiente heterogêneo de redes.

Aspectos relativos aos domínios de aplicação, arquitetura, funcionalidade e ambiente de desenvolvimento são fundamentais para a construção de uma plataforma e determinam tanto o seu ambiente de atuação, como a sua posterior admissão no mercado. Neste contexto, devem também ser observados os aspectos relativos às aplicações implementadas, aos serviços oferecidos, às ferramentas de desenvolvimento e demais requisitos de portabilidade e interoperabilidade.

Este capítulo apresenta um estudo dos diversos crítérios de avaliação considerando tópicos essenciais para o fornecimento de serviços de interface com o usuário, suporte a operações de gerenciamento, protocolos de comunicação e serviços de base de dados.

2.1 Critérios de Avaliação

Uma plataforma de gerência deve prover meios para a monitoração, controle e planejamento dos recursos existentes em uma rede e, em acréscimo, deve garantir que a introdução de tal funcionalidade não degrade o desempenho dos serviços fornecidos. Nesta análise, devem ser observados não somente o conjunto de funções existentes mas também, as ferramentas oferecidas aos usuários para o desenvolvimento de futuras extensões e os recursos de interoperabilidade com as demais plataformas de gerência existentes no mercado.

A atividade de avaliação consiste principalmente em identificar os componentes de uma plataforma de gerenciamento, suas capacidades funcionais, os recursos de projeto e operação e, em alguns casos, expor as restrições encontradas. Neste contexto, requisitos que incluem desde as aplicações oferecidas, o ambiente de gerenciamento e as ferramentas de desenvolvimento e implementação serão descritos neste trabalho. Aspectos adicionais relativos ao

fornecimento ou não do gerenciamento integrado, aberto ou distribuído, e demais características de engenharia de *software* associadas à modelagem das estruturas de informação, contribuem para a identificação dos componentes de uma plataforma [Ghet95].

A arquitetura de uma plataforma de gerenciamento pode ser constituída por meio de ferramentas e interfaces de programação, do ambiente de implementação e dos serviços de interface com o usuário, suporte a operações de gerenciamento, serviços de comunicação, manipulação de objetos e base de dados [Ghet95]. Tais aspectos serão detalhados nas próximas subseções.

2.2 Serviço de Interface com o Usuário

Atuando como suporte à apresentação das informações de gerenciamento, a interface com o usuário constitui o ponto de integração dos componentes da plataforma com as aplicações de gerenciamento. Fator determinístico para a operação de uma plataforma, as interfaces com o usuário geralmente seguem os padrões OSF/*Motif* e *XWindows* versão 11.

O serviço de interface com o usuário representa a principal área de investimento das empresas pretendem diferenciar seus sistemas de gerência com o objetivo de atrair novos compradores. Em geral, a interface com o usuário apresenta como requisitos funcionais as seguintes características:

- Representação gráfica da topologia da rede e demais componentes computacionais envolvidos;
- Interfaces para a realização de operações remotas;
- Divisão da área de operação em sub-domínios;

- Provisão de modos estatísticos para a avaliação de performance;
- Conformidade com padronizações (*Motif, X.11*); e
- Facilidades de reconfiguração da própria aplicação com o objetivo de suportar futuras extensões.

Em acréscimo, as interfaces gráficas também devem fornecer acesso alternativo a operações a partir de linhas de comando e, finalmente, garantir que as informações apresentadas correspondem ao estado atual dos recursos que estão sendo gerenciados.

2.3 Serviços de Suporte a Operações de Gerenciamento

O suporte a operações de gerenciamento oferece serviços comuns às diferentes aplicações de uma plataforma. Neste contexto, estão incluídos os processos encarregados da integração da plataforma com o sistema operacional subjacente, o relacionamento entre as aplicações de gerenciamento, a manipulação de sintaxes abstratas responsáveis pela representação das informações de gerenciamento e o fornecimento de mecanismos para a manipulação de eventos.

A gerência de redes é uma aplicação distribuída que envolve a troca de informações entre os processos de gerenciamento gerentes e agentes. Tal interação pode ser basicamente realizada pelas seguintes abordagens:

- Solicitação de uma operação por parte de um gerente a um agente através do emprego de primitivas de um protocolo de informação de gerenciamento (SNMP, CMIP ou algum protocolo proprietário);
- *Polling* (consulta seletiva), representando uma escrutinação periódica sobre os objetos; e

- Emissão de eventos que indicam a mudança de estado de algum recurso real.

Existem duas abordagens para organizar uma aplicação de gerenciamento com relação à comunicação entre os processos de aplicação. Uma abordagem síncrona, na qual cada solicitação é atendida imediatamente através do paradigma de execução concorrente; ou uma abordagem assíncrona, na qual as solicitações são escalonadas em uma fila, "a primeira solicitação que entra é a primeira a ser servida". Esta abordagem assíncrona é geralmente implantada por meio de um mecanismo de coordenação central que recebe os eventos externos e os distribui aos devidos objetos [PaBh93b].

Restrito não somente à modelagem e configuração de eventos, outro aspecto relevante consiste na correlação entre as notificações emitidas. Tal correlação objetiva reduzir o tráfego de informações redundantes e facilitar o diagnóstico de possíveis falhas que possam ocorrer em uma rede.

2.4 Serviços de Comunicação

Os protocolos de informação de gerenciamento são responsáveis pela comunicação entre os processos de gerenciamento e garantem a interoperabilidade dos provedores de serviços de gerenciamento distribuídos em um ambiente heterogêneo de redes.

Desta forma, surgiram no mercado padrões com o objetivo de possibilitar o gerenciamento de sistemas heterogêneos. Dentre estes padrões destacam-se o procotocolo de gerenciamento CMIP (*Common Management Information Protocol*) [IS 9596] proposto pela OSI (*Open Systems Interconnection*) e o protocolo SNMP (*Simple Network Management Protocol*) existente no modelo de gerência Internet. O

primeiro é um protocolo orientado a conexão e apresenta um escopo de operações capaz de realizar não somente a alteração e a consulta do valor de um objeto a ser gerenciado, como também possibilita que sejam realizadas ações sobre os objetos gerenciados. Em contraste, o protocolo SNMP, embora muito empregado no mercado, apresenta várias restrições funcionais em virtude de sua simplicidade de implementação [KoCo95a].

Os protocolos de gerenciamento estão intrinsicamente associados às interfaces para o desenvolvimento de aplicações de gerência. Estas interfaces objetivam ocultar completamente os detalhes de acesso ao serviço dos protocolos de gerenciamento relativos ao endereçamento das diversas entidades existentes em um sistema, e à manipulação da sintaxe ASN.1 (*Abstract Syntax Notation One*) [ISO 8824]. Em alguns casos, como os existentes para a arquitetura OSI, as interfaces para o desenvolvimento de aplicações de gerenciamento são responsáveis pelo escopo sobre níveis de hierarquia de um modelo de informação, filtragem de atributos e verificação de erros.

Amplamente adotada no mercado, a interface *XOpen XMP/XOM* tem se estabelecido como interface padrão comum aos protocolos CMIP e SNMP e exigida como requisito para as principais plataformas de gerenciamento.

2.5 Serviço de Manipulação de Objeto

Este serviço fornece suporte à troca de informação entre os objetos gerenciados. De acordo com [BRISA], para que seja garantida a interoperabilidade de diferentes sistemas de gerenciamento de redes, tais sistemas precisam ter uma visão comum da informação de gerenciamento a qual implica na definição de uma Estrutura de Informação de Gerenciamento SMI (*Structure of Management*

Information) que especifica o modelo de informação a ser adotado. Este modelo deve incluir a definição da estrutura de informação de gerenciamento armazenada em base de dados com este objetivo.

A modelagem de uma estrutura de informação irá garantir o suporte ao desenvolvimento de aplicações orientadas a objeto e a conseqüente definição das estruturas de dados. Disponível no mercado está uma grande variedade de padrões para a modelagem de objetos: *Internet Structure of Management Information* (SMI), *ISO OSI Management Information Model* (MIM), *OSF/DCE Object Model, Object Management Architecture* (OMA) e o *Microsoft Common Object Model* (COM) [Ghet95].

Em acréscimo, também são definidos alguns princípios para a definição de objetos tais como o padrão [ISO 10165-4] *Guidelines for the Definition of Managed Objects,* a Internet [RFC 1212] *Concise MIB Definitions* e a linguagem para definição de interface tal como o OMG IDL (*Object Management Group Interface Description Language*) [CORBA rv2.0].

2.6 Aplicações de Gerenciamento

Como mencionado anteriormente, as aplicações de gerenciamento são estruturadas por funções que atuam sobre as áreas de falhas, performance, configuração, contabilização e segurança.

Grande parte das plataformas de gerência concentram suas atividades no tratamento de falhas, medidas de configuração e ferramentas para análise de performance. Atividades de contabilização são principalmente empregadas em redes de telecomunicações enquanto que os mecanismos de segurança, anteriormente

desconsideradas, começam a adquirir relevância em virtude da crescente utilização das redes para fins comerciais, inclusive atividades bancárias.

O uso final da aplicação por parte do administrador da rede deve ocultar, entre outros, as restrições de acesso ao protocolo de informação, a manipulação das estruturas de informação e o mecanismo empregado para a coordenação de eventos. Ao usuário deve ser apresentado um módulo gráfico sobre o qual poderá realizar as operações de monitoração e controle e receber as devidas notificações.

Outro fator a ser considerado constitui a área de abrangência na qual está inserida a aplicação de gerenciamento. Está aplicação poderá atuar sobre uma rede local, metropolitana ou então como parte de uma rede de longa distância.

2.7 Ambiente de Desenvolvimento

Em adição às aplicações de gerência previamente implementadas, outro componente essencial em uma plataforma consiste no fornecimento de ferramentas para o desenvolvimento de novas aplicações de gerenciamento. Estas ferramentas permitem a construção de extensões que reflitam as necessidades específicas de cada usuário adminstrador de rede.

Neste conjunto de ferramentas estão incluidos:

- Blibliotecas de construções predefinidas e objetos gráficos;
- Editores para especificação de objetos GDMO e sintaxes ASN.1;
- Analisadores sintáticos;
- Compiladores de sintaxes abstratas para código de implementação;
- Interfaces para a programação de aplicações; e

- *MIB Browsers.*

O termo *MIB Browser* refere-se a aplicação gráfica fornecida ao usuário com o objetivo de possibilitar a pesquisa sobre uma base de informação de gerenciamento.

2.8 Análise de Performance

Paralelo ao estudo sobre os diversos compontes de uma plataforma é necessário observar a plataforma sobre o ponto de vista de alguns parâmetros de performance associados à carga de trabalho máxima e média suportada, ao tempo de resposta para obteção do resultado de uma operação, e ao comportamento diante de condições desfavoráveis.

Operações relativas à discriminação e relatório de eventos, ações de tomada de decisões, controle de tráfego e mecanismos automáticos de *backup* devem atuar de forma contínua.

3. Plataforma de Gerenciamento OSIMIS TMN

Desenvolvido pela *University College London*, este ambiente de gerenciamento denominado *OSI Management Information Service* (OSIMIS), tem como objetivo apresentar uma plataforma de gerenciamento genérica capaz de combinar toda a riqueza da funcionalidade OSI com a base, já instalada, de recursos *Internet* [PaBh93b].

Apesar de não ser um produto comercial, esforços foram feitos para que fossem empregados conceitos de engenharia de software com o objetivo de resultar em uma implementação eficiente. Desenvolvido por meio da abordagem orientada a objetos, o OSIMIS foi basicamente implementado em linguagens C e C++ e seu objetivo principal é fornecer um conjunto de APIs (*Application Program Interfaces*) para facilitar o desenvolvimento de aplicações de gerência. Estas APIs ocultam parte da complexidade do protocolo CMIP e possibilitam ao usuário do OSIMIS centralizar seus esforços muito mais na aplicação de gerência em desenvolvimento, do que nos mecanismos de acesso ao protocolo de informação [PaTi94].

Na realidade, a concepção do OSIMIS consiste no desenvolvimento de uma plataforma capaz de possibilitar a integração dos sistemas de gerenciamento existentes no mercado. A própria escolha da arquitetura OSI como modelo básico de gerenciamento facilita a integração com outras arquiteturas. Rico em funcionalidade, o modelo OSI apresenta facilidades que não são encontradas em modelos como o SNMP da Internet e o OMG CORBA (*Object Management Group Common Object Request Broker Architecture*).

Inserido nesta proposta de integração, o OSIMIS apresenta um *gateway* a nível da camada de aplicação entre os protocolos CMIP e SNMP, e esforços estão sendo dispendidos para a integração das arquiteturas OSI e OMG CORBA (*Object Management Group Common Object Request Broker Architecture*). Desta forma, o OSIMIS também entra em conformidade com a abordagem de organismos internacionais como o *Network Management Forum* e o *OMNIPoint* que sugerem uma arquitetura multiprotocolar para os futuros ambientes heterogêneos [KoCo95b].

Com o objetivo de possibilitar o desenvolvimento de aplicações de gerência o OSIMIS fornece os seguintes serviços:

- implementação do protocolo CMIP que utiliza os elementos de serviço ACSE (*Association Control Service Element*) e ROSE (*Remote Operations Service Element*) existentes no ISODE;
- implementação do protocolo SNMP;
- mecanismo de coordenação orientado a eventos responsável pela monitoração em tempo real das notificações emitidas pelos processos agentes;
- *Generic Managed System* (GMS) que representa um sistema gerenciado genérico, o qual é uma API para o desenvolvimento de novas classes de objetos gerenciados, e apresenta uma biblioteca com os atributos, notificações, objetos e funções de gerenciamento comumente utilizados;
- APIs para a construção de gerentes, estas APIs são denominadas de *Remote MIB* (RMIB) e *Shadow MIB* (SMIB);

- compiladores ASN.1 [RoOn91] e GDMO [GDMO93] que atuam, respectivamente, na construção de sintaxes de tipos de atributos em estruturas em linguagem C e na construção de objetos gerenciados.

- mecanismos de transparência de localização por meio de um suporte ao Serviço de Diretório X.500;

- *gateway* CMIP/SNMP que permite gerenciar objetos contidos em MIBs SNMP através do protocolo CMIP [PaBh93a] [SoMa93];

- conjunto de aplicações de gerenciamento genéricas (*MIB Browser*, *MIB Dump*, ...);

- agente para a versão OSI da MIB da camada TCP/IP; e

- agente para a camada de transporte OSI.

A organização dos componentes acima citados pode ser observada na figura 3.1. A parte em camada é formada por um conjunto de bibliotecas (GMS, RMIB, SMIB) que serve de interface para a construção de processos agentes e gerentes. Em acréscimo a estas bibliotecas, são encontrados os mecanismos de coordenação e os mecanismos de suporte à sintaxe ASN.1. O protocolo CMIP é implementado pelo OSIMIS, no entanto, a parte superior da pilha OSI até os elementos de serviço ACSE/ROSE é fornecido pelo ISODE. De forma análoga, a pilha Internet até o protocolo UDP (*User Datagram Protocol*) é fornecido pelo sistema operacional UNIX.

Na parte inferior, pode ser observado um conjunto de aplicações genéricas. Dentre estas aplicações, as ferramentas ASN.1 [RoOn91] e GDMO [GDMO93] são fundamentais para a construção de novas classes de objetos gerenciados. Estas ferramentas permitem que através de um arquivo fonte em sintaxe ASN.1 sejam

obtidas respectivamente, estruturas e códigos em linguagem C e C++. O *gateway* CMIP/SNMP este consiste no mapeamento de primitivas CMIP para as correspondentes primitivas SNMP. Desta forma, gerentes CMIP conseguem manipular MIBs SNMP.

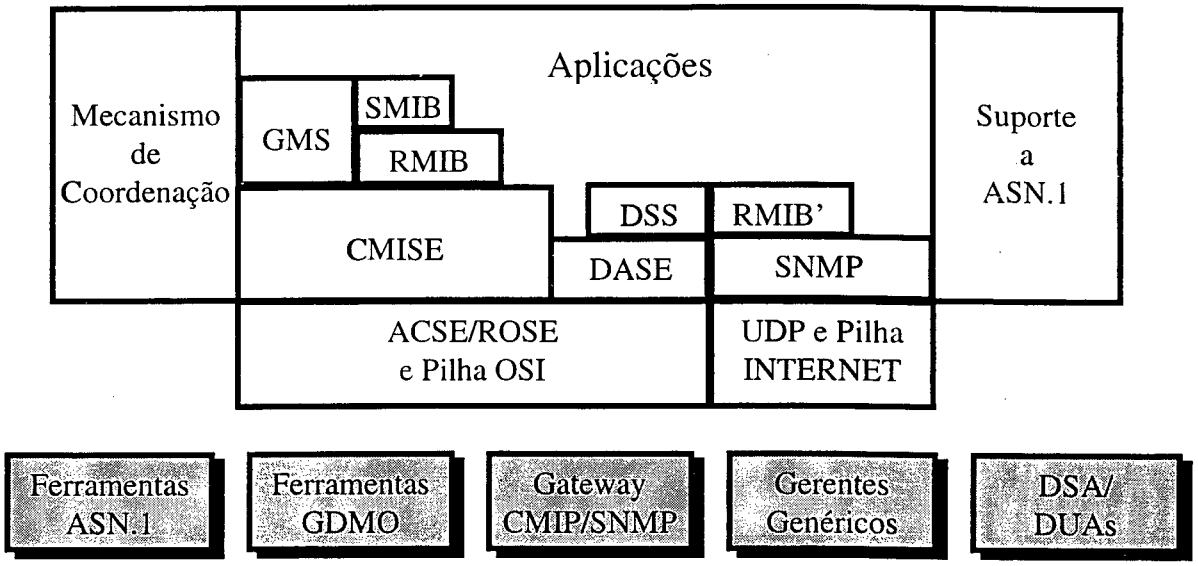

Figura 3.1: Arquitetura da plataforma de gerenciamento OSIMIS [PaKn95].

Entre os componentes apresentados na figura 3.1, destacam-se os mecanismos de coordenação e de suporte a manipulação da sintaxe abstrata ASN.1, as APIs para o desenvolvimento de processos de gerência e algumas aplicações de gerenciamento genéricas.

3.1.1 Suporte a Aplicações Assíncronas Orientadas a Eventos

Gerência de redes é uma aplicação distribuída que envolve a troca de informações entre os processos de gerenciamento e os objetos gerenciados. A interação destes processos de gerenciamento com os objetos gerenciados pode-se realizar de acordo com três abordagens:

- acesso por solicitação externa através do emprego das primitivas CMIP;

- *polling*, representando uma consulta periódica sobre os objetos gerenciados; e
- através de relatórios de eventos que indicam a alteração no estado de algum recurso real.

Com referência aos itens anteriores, deve-se prover meios para que gerentes se comuniquem com os agentes, os agentes devem se comunicar não somente com os gerentes, mas também, com os objetos gerenciados. Há duas maneiras de organizar uma aplicação de gerenciamento com relação a comunicação entre os processos gerente, agente e os objetos gerenciados: a) uma abordagem síncrona, onde cada solicitação é atendida através do paradigma de execução concorrente; ou b) uma abordagem assíncrona, onde as solicitações são escalonadas em uma fila, "o primeiro que entra é o primeiro a ser servido". Esta abordagem assíncrona pode ser implantada através de um mecanismo de coordenação central que recebe não somente os eventos externos, como também, os eventos escalonados em relação aos recursos reais. Das duas abordagens mencionadas anteriormente, restrições como: possibilidade de ocorrência de impasses (*deadlocks*), dificuldade de depuração de aplicações concorrentes em tempo real e, principalmente, a ausência de alguns mecanismos de concorrência por parte dos sistemas operacionais comumente utilizados, tornaram a abordagem síncrona inviável [KoCoS4b].

Desta forma, o OSIMIS fornece uma estrutura orientada a objetos que permite a organização de aplicações em um regime totalmente orientado a eventos. Em conformidade com a abordagem assíncrona, cada evento interno ou externo é escalonado em uma fila. Este mecanismo prevê a integração com outros mecanismos de coordenação e é implementado através de duas classes de objetos em linguagem C++: *Coordinator* e *Knowledge-Source(KS)*.

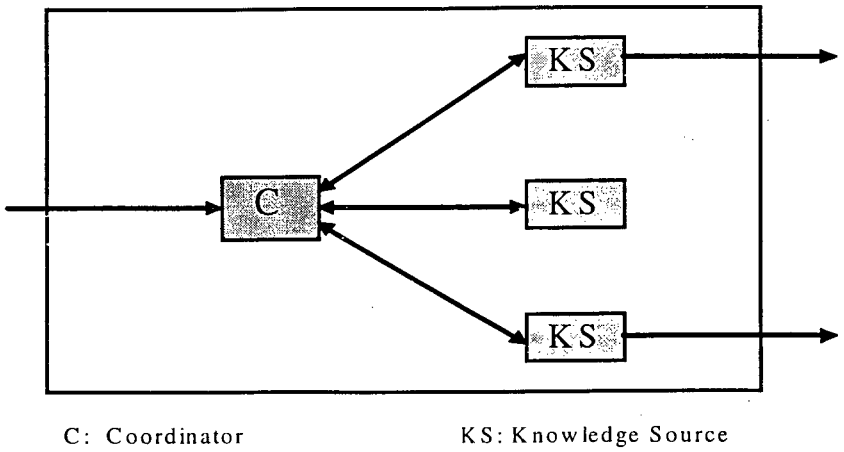

Figura 3.2: Mecanismo de coordenação orientado a eventos [PaKn95].

Como pode ser observado na figura 3.2, a classe *KS* é responsável pela coleta de eventos no sistema e, por seguinte, repasse destes eventos para uma instância da classe *Coordinator*. Esta classe também atua na definição da estratégia de *polling* a ser utilizada para a monitoração dos objetos gerenciados. A classe *Coordinator* recebe os dados de KS e os repassa ao gerente responsável.

3.1.1.1 A Classe *Knowledge-Source* (KS)

Esta classe deve prover o escalonamento de alarmes (*wake-ups*) em intervalos de tempo regulares com o objetivo de implementar o regime de *polling*. A classe KS também atua no registro de comunicações externas tanto para os processos de gerenciamento, como para os recursos reais que aguardam receber informações de gerenciamento.

A classe *Knowledge-Source* pode ser utilizada tanto pelos agentes como pelos gerentes. Para os processos gerentes a classe é responsável por toda a coleta e recebimento da informação. Enquanto que, para os processos agentes ela é utilizada tanto para a definição da estratégia de *polling*, como para o recebimento de

informações dos recursos que não estão em um espaço de endereçamento comum, como por exemplo, outros agentes ou recursos remotos.

3.1.1.2 A Classe *Coordinator*

Em virtude da elevada quantidade de informações que trafegam em um sistema de gerência, foi necessário o desenvolvimento de uma mecanismo que garantisse que estas informações fossem direcionadas para os devidos objetos. Neste contexto, o classe *Coordinator* coordena as atividades das aplicações de gerenciamento atuando como ponto central no recebimento de comunicações externas, como também, no escalonamento de alarmes (*wake-ups*). Desta forma, todas as mensagens emitidas são recebidas pelo *Coordinator* que as distribui aos devidos objetos.

Através de algumas ferramentas existentes no UNIX, esta classe implementa um esquema totalmente direcionado a eventos usando uma política assíncrona de filas que regula as atividades relacionadas com a comunicação externa e o escalonamento de alarmes.

Esta classe possui métodos que permitem a integração com outras aplicações que possuem seu próprio mecanismo de controle. Neste caso, o mecanismo de controle da aplicação juntamente com derivações da classe *Coordinator* que centralizarão as atividades de comunicação.

3.1.2 Suporte a Manipulação da Sintaxe ASN.1 para a Definição dos Dados

As estruturas de dados utilizadas no modelo de gerenciamento OSI são representadas em sintaxe abstrata ASN.1, no entando, durante a construção de aplicações de gerenciamento estas estruturas de dados devem ser transformadas em estruturas correspondentes em linguagem de programação.

O OSIMIS fornece um mecanismo de suporte à sintaxe ASN.1 que tem como objetivo impedir o implementador de manipular estruturas de dados abstratas. Desta forma, os tipos de dados não são representados por estruturas ASN.1, mas por objetos em linguagem C++. Estes objetos contém métodos polimórficos que, entre outras funções, permitem que dados sejam codificados e decodificados em estruturas C e ASN.1. Este mecanismo de suporte a sintaxe ASN.1 está disponível no OSIMIS por meio de duas classes: *Attribute* e *anyType*.

3.1.2.1 **A Classe *Attribute***

A classe *Attribute* é a superclasse de todos os atributos de gerenciamento. Como característica importante, esta classe utiliza a estrutura de dados da linguagem C em virtude do compilador ISODE ASN.1 não manipular estruturas C++.

Como o OSIMIS estabelece que cada tipo de atributo deve ser definido como uma classe derivada de *Attribute*, alguns métodos desta classe podem ser redefinidos para expressar melhor a sintaxe deste novo atributo.

Dentro do próprio sistema de gerenciamento genérico do OSIMIS algumas classes de tipos de atributos comumente utilizados em gerenciamento são previamente fornecidas para uso imediato do implementador de aplicações. Desta forma, alguns atributos como: *integer, real, string, counter, gauge, administrative state* e *operational state* são todas classes já existentes no OSIMIS. No entanto, a implementação de um novo objeto gerenciado poderá resultar na necessidade da definição de um novo tipo de atributo ainda não existente que deve ser, então, implementado como uma subclasse de *Attribute*.

3.1.2.2 A Classe *anyType*

O OSIMIS utiliza ferramentas fornecidas pelo ISODE para a manipulação da sintaxe *ASN.1* durante a construção das tabelas de conversão. A classe *Attribute* foi implementada para automatizar esta conversão, evitando desta forma o emprego de tabelas. Tal abordagem resulta em uma sobrecarga de operação em virtude de que toda vez que um atributo é instanciado a tabela deve ser consultada.

A classe *anyType* é uma subclasse de *Attribute* que provê a manipulação sintática através de tabelas de forma não automática. Esta classe é utilizada quando não se deseja refinar os métodos de manipulação sintática fornecidos pela classe *Attribute*, casos nos quais a conversão dos dados por meio de tabelas for mais simples.

3.1.3 Interfaces para a construção de processos gerentes

Para facilitar a construção de processos gerentes são fornecidas pelo OSIMIS APIs com este objetivo. Estas APIs, *Remote MIB* e *Shadow MIB*, são descritas a seguir.

3.1.3.1 *Remote MIB*

Em virtude do protocolo CMIP possuir certa complexidade, e as APIs até então desenvolvidas não apresentarem facilidade de utilização, era mantida sob responsabilidade do implementador tarefas adicionais aos requisitos da aplicação de gerência. Como exemplo, para a realização de funções de escopo e filtragem é necessário conhecer toda a estrutura de dados dos objetos. Para a identificação de classes de objetos, atributos, ações, relatório de eventos é necessário o uso de identificadores de objetos (*Object Identifiers* - OIDs). Em acréscimo a estas restrições apresenta-se o fato de que somente uma informação de instância de um

objeto gerenciado pode ser transportada em uma unidade de dados de protocolo (*Protocol Data Unit* - PDU) de resposta [PaTi94]. Em notificações onde pode-se fornecer como resposta um ou mais objetos gerenciados, cabe a quem está desenvolvendo a aplicação receber todas as PDUs e implementar meios necessários para compor o quadro de informações.

A *Remote MIB* (RMIB) é o resultado de pesquisas que objetivam solucionar estas restrições com o desenvolvimento de APIs de alto nível. Estas APIs oferecem uma interface ao protocolo CMIP que ocultam ao máximo os parâmetros de comunicação. A *Remote MIB* (RMIB) utiliza para a nomeação de classes, objetos, atributos, e relatórios de eventos uma notação baseada em strings. O emprego de *strings* facilita a construção e a interpretação das operações de escopo e filtragem. Para obter os endereços reais dos elementos da rede os *strings* são mapeados, por meio de tabelas, para os octetos definidos no modelo OSI.

Certas operações fornecem como resposta o valor dos atributos de vários objetos gerenciados. Devido à restrição do protocolo CMIP de que somente uma informação de instância é transportada em cada PDU, as respostas destas operações são compostas por várias PDUs. A RMIB fornece dois meios para facilitar a obtenção das respostas destas operações:

- os resultados são passados um a um, onde cada PDU é uma instância da classe *CMISObject* ;
- todos passados coletivamente dentro de apenas uma unidade, implementado através da classe *CMISObjectList*.

A estrutura genérica de funcionamento da interface RMIB é descrita na Figura 3.3.

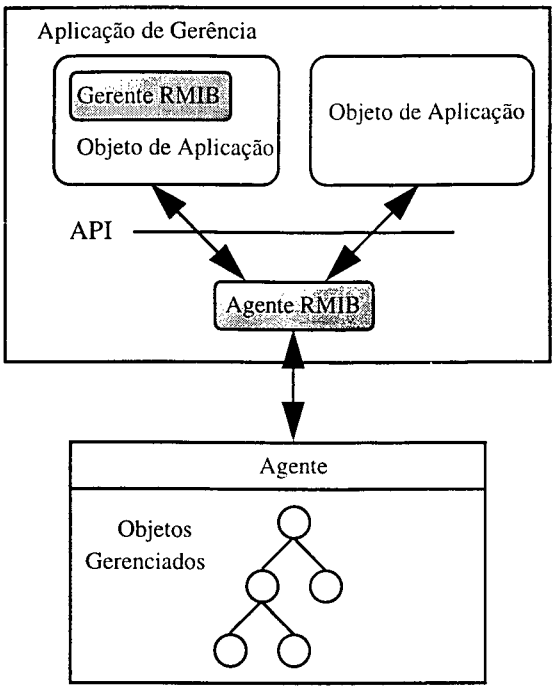

Figura 3.3: A Interface RMIB [PaTi94].

A RMIB é implementada através da classe *RMIBAgent*, cada instância desta classe está associada a um agente remoto. Em acréscimo ao controle de associação é oferecida uma interface para o acesso ao protocolo CMIP. Com este recurso as mensagens não são mais enviadas diretamente para o agente remoto, mas para o agente local RMIB que repassa as mensagens ao gerente.

3.1.3.2 Shadow MIB

Uma das atividades mais comuns em qualquer aplicação de gerência consiste no acesso a informações sobre os objetos gerenciados. A freqüência de acesso a esses dados depende do tipo de dado e da necessidade atual da aplicação. Por exemplo, quando se deseja monitorar o desempenho da rede, o número de pacotes que trafegam pelos roteadores é constantemente monitorado. A freqüência com que os dados se alteram é outra característica a ser observada por uma aplicação de gerência [PaTi94].

Considerando tais características deve-se decidir qual o método mais adequado para o acesso a cada informação na rede. Este procedimento deve ser adotado para reduzir a comunicação entre gerente e agente ao mínimo necessário, pois uma vez que se adote uma decisão errada, essa comunicação será realizada desnecessariamente. Em consequência o *software* de gerência poderá contribuir para uma redução no desempenho na rede.

Nos casos onde os dados se alteram mais frequentemente do que o gerente os acessa deve ser utilizado um mecanismo de *polling* para o acesso à informação. Essa escolha é justificada pelo fato de que somente ocorrerá troca de informações de gerência quando o gerente necessitar dos dados, e não a cada vez que os dados se alteram.

Em contrapartida, quando os dados se alteram poucas vezes em determinado intervalo de tempo, e neste mesmo intervalo a informação é monitorada muitas vezes, um mecanismo orientado a eventos deve ser utilizado. Desta forma, a cada alteração de dados o gerente será informado. Como, nesta situação, as alterações têm freqüência menor que o acesso a elas, novamente o tráfego de informações de gerência será reduzido ao mínimo necessário. Assim, para realizar o acesso a informações o gerente pode possuir um *cache* local que refiita os dados dos objetos na rede. Nos caso de alteração o gerente deve acessar o agente, uma vez que tanto o objeto gerenciado quanto o *cache* devem ser atualizados. A vantagem deste método é ainda mais visível quando o gerente consulta os dados muito mais vezes do que os altera.

Para a realização deste acesso aos dados torna-se necessário definir uma interface para facilitar a sua implementação. Justifica-se então os estudos relativos à API *Shadow MIB* (SMIB). Por meio desta interface são implementados mecanismos

de *cache* com o objetivo de atender as necessidades de cada aplicação de gerência. Para os casos onde é impossível utilizar um mecanismo orientado a eventos para a obtenção dos dados, a SMIB fornece um mecanismo de *polling*.

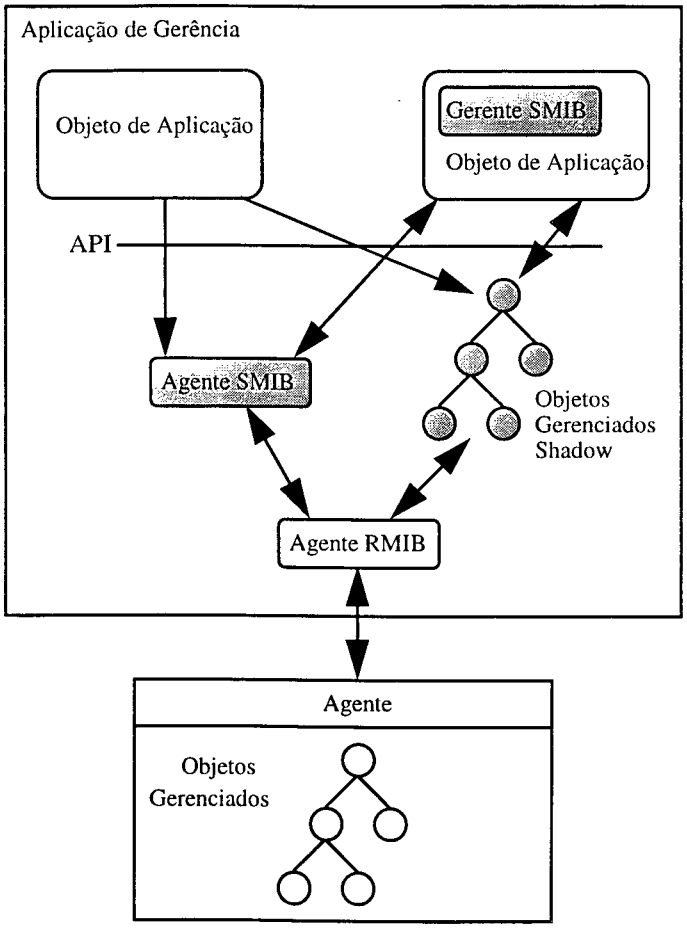

Figura 3.4: A Interface SMIB [PaTi94].

A Figura 3.4 descreve como a interface SMIB está estruturada. A sua funcionalidade torna-se disponível através do uso de duas classes, *SMIBAgent* e *SMO* (*Shadow Managed Object*). Operações sobre uma sub-árvore específica da MIB são feitas diretamente sobre uma instância de *SMO*, outras operações são feitas através de intâncias da API *SMIBAgent*.

3.1.4 Interface para a construção de processos agentes

3.1.4.1 GMS (*Generic Managed System*)

Tem como objetivo principal auxiliar a construção de agentes e facilitar, em conjunto com o compilador GDMO, a construção e a integração de objetos gerenciados [KoCo94b].

Para facilitar a construção de agentes e objetos gerenciados o GMS procura ocultar completamente os detalhes de acesso a informação do protocolo CMIP. Desta forma tarefas como endereçamento de objetos, escopo sobre níveis de herança, filtragem e verificação de erros saem do encargo do desenvolvedor da aplicação.

Estas funções tornam-se disponíveis através de APIs, implementadas nas quatro classes que compõem o GMS:

- Classe *CMISAgent*;
- Classe *MO*;
- Classe *MOClassInfo*;
- Classe *Top*.

3.1.4.2 Classe *CMISAgent*

Trata-se de um agente *Knowledge-Source* (derivado da classe KS), responsável pela captação de eventos provenientes do ISODE. Este agente atua no estabelecimento de conexões de gerenciamento sendo responsável em aceitar ou recusar estas conexões através de autenticação de informações e validação dos parâmetros de comunicação. Em acréscimo a estas tarefas agente também realiza o endereçamento, tarefas de escopo e filtragem, além de retornar o resultado de uma operação. Cada aplicação no papel de agente possui um instância desta classe,

uma vez que a utilização desta classe facilita a implementação, e a realização dos serviços de conexão.

3.1.4.3 Classe *MO*

A classe MO é superclasse de todas as classes de objetos gerenciados e fornece suporte para a construção de objetos gerenciados genéricos. Nesta classe são implementados métodos que encapsulam parte da complexidade do protocolo CMIP. Através da utilização dos métodos disponíveis nesta classe facilita-se o trabalho de acesso e consulta as bases de informação de gerenciamento.

3.1.4.4 Classe *MOClassInfo*

O principal motivo para a existência desta classe é fornecer um modelo para construção de classes de objetos gerenciados. Ela contém identificadores de atributos, grupo, evento e ação. Contém também um ponteiro para a primeira instância da classe, e através desta as outras instâncias serão endereçadas.

3.1.4.5 Classe *Top*

Consiste na raiz da Árvore de Informação de Gerenciamento - *Management Information Tree* (MIT). A descrição do método de acesso utilizado pelos processos para acessar cada objeto gerenciado está contida nesta classe. Em acréscimo fornece uma interface para a função de relatório de eventos.

3.1.5 Compilador GDMO

Como mencionado anteriormente, o modelo de gerenciamento OSI utiliza-se do paradigma de orientação a objetos para representar as diversas entidades envolvidas em um sistema. Cada entidade a ser monitorada é representada por um ou mais objetos que são denominados de "objetos gerenciados". Para a

representação formal destes objetos pode ser utilizada a linguagem GDMO (*Guidelines for the Definition of Managed Objects*) [ISO 10165-4]. Esta linguagem, embora independente da implementação, fornece a sintaxe necessária para especificação de todos os componentes que definem um objeto gerenciado[1].

Utilizando-se dos conceitos de especificação em linguagem abstrada, o OSIMIS oferece uma ferramenta para a construção de novas classes de objetos gerenciados. Esta ferramenta, identificada como o compilador GDMO, é utilizada para a geração de classes de objetos gerenciados C++ a partir de um código fonte especificado em GDMO.

A relevância da utilização deste compilador pode ser observada em virtude de grande parte das normalizações propostas no modelo de gerenciamento OSI estarem especificadas em GDMO. Parte do trabalho de implementação pode ser facilitado através da conversão automatizada de especificações GDMO para a linguagem C++.

O funcionamento do compilador pode ser representado através de quatro *scripts*. O primeiro controla todo o trabalho de geração de código, o segundo é responsável em criar o arquivo *Makefile* que será utilizado para criar as bibliotecas de objetos gerenciados. Por fim, são apresentados os dois *scripts* empregados na construção dos arquivos de *header* (definição) e de métodos necessários à implementação das classes de objetos gerenciados C++.

---

[1] Um objeto no modelo de gerenciamento OSI pode ser definido através de seus **atributos**, que contêm informações de interesse na representação de um recurso; do **comportamento** de acordo com as operações recebidas; das **notificações** indicando algum evento ocorrido; e das **operações** que podem ser efetuadas sobre o atributo.

3.1.6 Gateway CMIP/SNMP a Nível da Camada de Aplicação

De acordo com [PaBh93a] a coexistência dos protocolos de gerenciamento CMIP e SNMP sugere três abordagens distintas: a) uso de gerentes que adotem ambos os modelos OSI e Internet e utilize determinado protocolo de gerenciamento de acordo com os requisitos de comunicação. Tal abordagem apresenta restrições em virtude de concentrar nos processos gerentes a complexidade da atividade de gerenciamento e demandar a presença de duas pilhas de protocolos OSI e Internet. b) uso de agentes duplos no qual é realizada a reimplementação de um agente SNMP para um agente OSI; e o c) emprego de um *gateway* a nível da camada de aplicação para a devida tradução de operações CMIP em correspondentes SNMP e a devida construção de notificações a partir de *traps* provenientes de agentes SNMP.

O conceito utilizado na plataforma OSIMIS consiste no uso de um agente OSI que atua como um *gateway* realizando acesso a um agente SNMP localizado em um ponto terminal remoto. Sob a perspectiva da arquitetura TMN, a atividade de tradução fornecida pelo *gateway* pode-se configurar como uma função de adaptador Q para a interface com elementos não TMN. O funcionamento do *gateway* pode ser caracterizado em três fases distintas que incluem inicialmente a tradução de uma especificação ASN.1 de uma base de informação SNMP para uma correspondente MIB CMIP, a conversão de uma especificação GDMO em um arquivo de descrição da MIB e finalmente, o mapeamento de operações CMIP para as respectivas operações SNMP.

3.1.7 Aplicações de Gerenciamento

A plataforma OSIMIS fornece aplicações que permitem, não somente, a monitoração dos agentes, como também, a requisição de registro de eventos, a

manipulação das tabelas de roteamento IP, e a utilização de uma interface que possibilita a pesquisa através da Árvore de Informação de Gerenciamento MIT.

Encontram-se a seguir, as aplicações de gerenciamento fornecidas pelo OSIMIS:

Um conjunto de programas que permite que operações, por meio dos serviços CMIS, sejam realizadas sobre os agentes. Estas operações acarretarão em ações sobre os objetos gerenciados[KoCo94b].

- mcreate: Este programa estabelece uma conexão com um agente remoto e cria um objeto gerenciado em sua respectiva MIB. Inicialmente, é estabelecida uma associação de gerenciamento; em seguida, a operação *create* é solicitada sendo executada de acordo com os argumentos especificados. Após a emissão dos resultados a associação é liberada.

- mset: Este programa estabelece uma conexão com um agente remoto com o objetivo de alterar os atributos de um objeto gerenciado. Inicialmente, é estabelecida uma associação de gerenciamento; em seguida, é solicitada a operação *set* que, de acordo com as diferentes opções, pode alterar, adicionar ou remover elementos em atributos definidos como conjunto, e ainda alterar um atributo para o seu valor *default*. Finalmente é emitido o resultado e a associação liberada.

- maction: Este programa estabelece uma conexão com um agente remoto com o objetivo de realizar uma ação sobre o objeto gerenciado. De forma análoga as anteriores, é estabelecida uma operação de gerenciamento, em seguida uma operação *action* é efetuada sendo os resultados emitidos e a associação liberada.

- mdelete: Um programa que permite a conexão com um agente remoto com o objetivo de remover um objeto gerenciado de sua respectiva MIB.

- mibdump: Um programa que permite a conexão com um agente remoto com o objetivo de consultar os valores dos atributos dos objetos gerenciados especificados. Após coletada as informações a associação é liberada .

Dois programas que permitem que notificações sejam solicitadas, recebidas ou armazenadas:

- evsink: Através da conexão com um agente, este programa solicita a coleta dos relatórios de eventos de uma MIB, emitindo continuamente os resultados atualizados destes relatórios.

- evlog: Implementa a função de registro de *log* descrito pela OSI, na recomendação [ISO 10164-6]. *ISO 10164-6: "Information Technology - Open Systems Interconnection - Systems Management Functions - Part6: Log Control Function".*

Uma ferramenta para a monitoração da Base de Informação de Gerenciamento(MIB).

- cmisbrowser: Consiste de uma interface gráfica *XWindows* que permite ao usuário consultar as instâncias de objeto existentes em uma MIB [PaCo92]. O *cmisbrowser* é uma aplicação genérica e pode ser aplicada a qualquer agente CMIS sem o conhecimento prévio da estrutura da árvore de gerenciamento. Esta ferramenta permite realizar também a alteração de atributos e a coleta de informações mais atualizadas dos estados dos objetos presentes na MIB.

3.1.8 As MIBs Fornecidas

O OSIMIS versão 4.0 apresenta as seguintes Bases de Informação de Gerenciamento:

- ISODE_MIB: utilizada no gerenciamento da camada de transporte implementada no ISODE. Apesar de não seguir o padrão OSI, os objetos gerenciados são os mesmos especificados na norma ISO/IEC 10737 "Gerenciamento da Camada de Transporte".
- UX_MIB: um exemplo simples de uma MIB sobre a função *users* existente no UNIX, cujo único objetivo é relatar o número de usuários utilizando o sistema operacional.
- OIM_MIB: uma implementação de uma Base de Informação de Gerenciamento OSI - Internet, uma visão OSI das camadas TCP/IP.

3.1.9 Os Agentes Implementados

Com o objetivo de demonstrar algumas das características desta plataforma de gerenciamento, tais como: as diferentes formas de acesso aos recursos reais, emprego dos serviços CMIS e a comunicação com os gerentes e objetos gerenciados, o OSIMIS fornece dois agentes que servem de exemplo para os usuários do GMS.

- SMA: Um agente que implementa uma MIB para o Protocolo de Transporte definida pela ISO, usado na implementação do ISODE. Este agente também implementa uma MIB referente ao sistema operacional UNIX, contendo objetos gerenciados cujo único objetivo é mostrar o número de usuários no sistema.

- OIMSMA: Um agente que implementa a MIB para integração OSI-Internet, como definido pela norma RFC 1214: "Gerenciamento OSI Internet: Base de Informação de Gerenciamento".

4. Extensões para a Plataforma de Gerência OSIMIS

Com o objetivo de explorar as interfaces para a construção de processos de gerenciamento existentes na plataforma OSIMIS e demais ferramentas para a construção de aplicações de gerência (i.e. compiladores ASN.1 e GDMO) foram desenvolvidas extensões à plataforma [KoCo94a]. A implementação destes novos serviços envolveu o emprego de parte dos componentes fornecidos pelo OSIMIS e forneceu subsídios para o posterior desenvolvimento de aplicações de gerência.

Como observado no capítulo anterior, a plataforma OSIMIS está em conformidade com a arquitetura OSI e portanto, o desenvolvimento de extensões está intrinsecamente associado aos requisitos do paradigma de orientação a objetos e aos métodos de acesso ao protocolo CMIP. No entanto, a metodologia para a construção de aplicações de gerenciamento sofreu adaptações com o objetivo de atender às interfaces de programação e às ferramentas fornecidas pela plataforma. Componentes distintos como as interfaces para a construção de processos gerentes e agentes, e os compiladores para a manipulação das sintaxes ASN.1 e GDMO foram incluídos nos procedimentos para a construção de aplicações de gerência.

Em contrapartida, componentes isolados das aplicações em desenvolvimento exigem tratamento específico e são construídos pelo emprego de diferentes recursos da plataforma OSIMIS. Como exemplo, os mecanismos utilizados para a manipulação da sintaxe abstrata ASN.1 independem dos métodos de coordenação de eventos que serão posteriormente utilizados durante a comunicação entre os objetos gerenciados. Em consequência, as abordagens de projeto e implementação são específicas para cada fase do desenvolvimento de aplicações de gerência e identificam os recursos da plataforma a serem utilizados.

4.1 Método para o desenvolvimento de extensões à plataforma OSIMIS

O modelo de gerenciamento OSI utiliza os conceitos de orientação a objetos. Desta forma, os recursos gerenciados são representados logicamente nos sistemas de gerenciamento por meio de entidades denominadas objetos gerenciados. Um objeto gerenciado pode ser definido através de: a) seus atributos, que contêm informações relevantes para a representação de um recurso; b) operações que podem ser efetuadas sobre o objeto; c) seu comportamento diante das requisições de operações que o objeto deve executar; e d) notificações que indicam a ocorrência de algum evento. Tais componentes podem ser especificados por meio de estruturas em linguagem ASN.1 as quais descrevem os princípios para a definição de objetos gerenciados GDMO [ISO 10165-4].

A primeira etapa para a construção de novas aplicações de gerência, consiste na identificação dos recursos físicos ou lógicos a serem modelados e a posterior especificação destes objetos em estruturas GDMO.

Aplicações de gerência manipulam uma elevada quantidade de informações, em consequência, parte dos esforços para a implementação de objetos gerenciados está associada a construção de atributos. Estes atributos podem apresentar uma estrutura complexa a qual envolve, entre outros, o emprego de tipos abstratos de dados como listas, sequências e conjuntos. Em acréscimo, para a construção de um novo tipo de atributo devem ser observados não somente os aspectos sintáticos associados à sua estrutura, como também os aspectos semânticos que definirão o seu comportamento.

Após especificado o objeto gerenciado e implementada a parte sintática de seus atributos e notificações, o código obtido será utilizado pelo compilador GDMO

para a construção parcial em linguagem C++ da classe de um objeto gerenciado. Os arquivos de definição e implementação gerados pelo compilador deverão ser incrementados com o código referente ao comportamento do objeto. Uma vez concluído o objeto, este deverá ser integrado ao agente ao qual estará subordinado .

A construção de agentes para a plataforma OSIMIS é favorecida pelo emprego da interface GMS a qual contém, previamente implementados, componentes comumente utilizados em um agente. Esta interface realiza a codificação e decodificação dos pacotes de dados que são enviados e recebidos ao agente, trata das solicitações CMIP, coordena a comunicação de eventos entre objetos gerenciados e manipula a MIB [PaBh93b].

Em resumo, as principais etapas envolvidas no processo de desenvolvimento de uma aplicação de gerenciamento podem ser discriminadas como:

1. Identificação dos recursos físicos e lógicos a serem gerenciados;
2. Especificação ASN.1 da sintaxe dos atributos e notificações;
3. Utilização do compilador *Pepsy* para a geração automática de estruturas em linguagem C;
4. Desenvolvimento de funções para manipulação de sintaxe;
5. Construção de classes de tipos de atributos;
6. Especificação de um objeto gerenciado em GDMO: atributos, operações, notificações e comportamento;
7. Utilização do compilador GDMO para a geração automática de estruturas em linguagem C++;
8. Inclusão dos aspectos semânticos das ações e notificações no código gerado pelo compilador GDMO;
9. Integração do objeto gerenciado em um agente; e,

10. Utilização das interfaces SMIB e RMIB para a implementação de processos gerentes.

A descrição detalhada destas etapas será apresentada nas subseções a seguir.

4.1.1 Especificação ASN.1 da sintaxe dos atributos e notificações

No caso da plataforma OSIMIS, os aspectos sintáticos e semânticos dos atributos são tratados separadamente. A partir de uma especificação ASN.1 da sintaxe de um atributo deve ser obtida a correspondente representação em linguagem de programação. O compilador *Pepsy* [RoOn91] realiza esta atividade de conversão e constrói tipos de dados em linguagem C. Como resultado do processo de compilação, dois arquivos são gerados: um arquivo de definição (*header*) e um arquivo de implementação. O primeiro define as estruturas de dados em linguagem C para os valores associados a cada sintaxe de atributo e declara as funções de codificação e decodificação implementadas para cada atributo. O segundo arquivo gerado contém a implementação das funções e de tabelas que realizam o mapeamento automático de C para ASN.1[KoCo95a] [KoCo95b].

O compilador *Pepsy* deve ser configurado para gerar as seguintes funções para cada sintaxe de atributo:

- *Build*: gera um elemento de apresentação (*Presentation Element*) com o valor do atributo em sintaxe ASN.1;

- *Parse*: realiza a operação inversa. Retorna uma estrutura em C a partir de um elemento de apresentação ASN.1.

- *Print*: imprime os componentes de um elemento de apresentação em uma cadeia de caracteres.

Estas funções são necessárias para que os processos de gerência troquem informações em um formato padronizado, no caso, a sintaxe ASN.1. O emprego destas rotinas possibilitam que projetistas de aplicações de gerência fiquem isentos dos detalhes associados à manipulação da sintaxe abstrata. O mapeamento de estruturas C para elementos de apresentação ASN.1 é mantido sob atuação do compilador *Pepsy*.

4.1.2 Funções para Manipulação de Sintaxes de Tipos de Atributos

Os processos de gerência necessitam ter acesso a funções para manipulação dos valores dos atributos que compõem os objetos gerenciados. Para cada sintaxe de atributo implementada, as seguintes funções devem ser providas:

- *Encode*: invoca a função *build*, gerada pelo compilador *Pepsy;*
- *Decode*: invoca a função *parse*, gerada pelo compilador *Pepsy*;
- *Print*: retorna o valor do atributo na forma de uma cadeia de caracteres;
- *Parse*: constrói o valor de um atributo a partir de uma cadeia de caracteres;
- *Compare*: compara dois valores de atributos, retornando o valor 0 no caso dos valores serem iguais; -1 quando o primeiro valor for menor; ou 1 em situações nas quais o primeiro valor for maior.

Outras funções devem ser implementadas, dependendo das características da sintaxe dos tipos de atributo. Para atributos multi-valorados, por exemplo, devem ser providas as funções *getelem* e *getnext*, que atuam respectivamente, para retornar o valor ou o sucessor do elemento de uma lista.

4.1.3 Implementação da Classe de Atributo

As etapas apresentadas até o momento têm como objetivo a implementação dos aspectos relacionados exclusivamente à sintaxe de um atributo. Aspectos relacionados à semântica do atributo, tais como seu comportamento, são implementados em uma classe de objeto. Cada classe será efetivamente utilizada para a composição dos objetos gerenciados e para a construção de outras classes de atributos.

Uma classe de atributos deve fornecer o seguinte comportamento:

- métodos para o atendimento das primitivas orientadas a atributos do serviço CMIS, tais como *M-Get* e *M-Set*;

- métodos para codificação e decodificação do valor associado ao atributo;

- métodos que permitam que o valor do atributo reflita o estado corrente de alguma propriedade do recurso real sendo representado.

Métodos genéricos que fornecem a funcionalidade mínima requerida para uma classe de atributos são fornecidos por mecanismos de suporte a manipulação de sintaxe abstrata existente na plataforma OSIMIS. Como observado no capítulo anterior, este mecanismo é formado pelas classes de objeto *Attribute* e *anyType*. No caso da implementação de uma nova classe de atributos, esta deve herdar direta ou indiretamente métodos de *Attribute*. Em acréscimo, estes métodos devem ser especializados com o objetivo de introduzir as características particulares de cada classe.

4.1.3.1 Diagrama de Implementação de atributos

Os métodos utilizados para a implementação de novos tipos de atributos podem ser observados na figura 4.1. Inicialmente, são enviados ao compilador *Pepsy* as estruturas em linguagem ASN.1 relativas as sintaxes dos tipos de atributos a serem implementados. Após o processo de compilação são gerados arquivos contendo funções e tabelas para codificação e decodificação de estruturas C e ASN.1. Estas tabelas e funções são utilizadas na implementação das funções para manipulação de sintaxe.

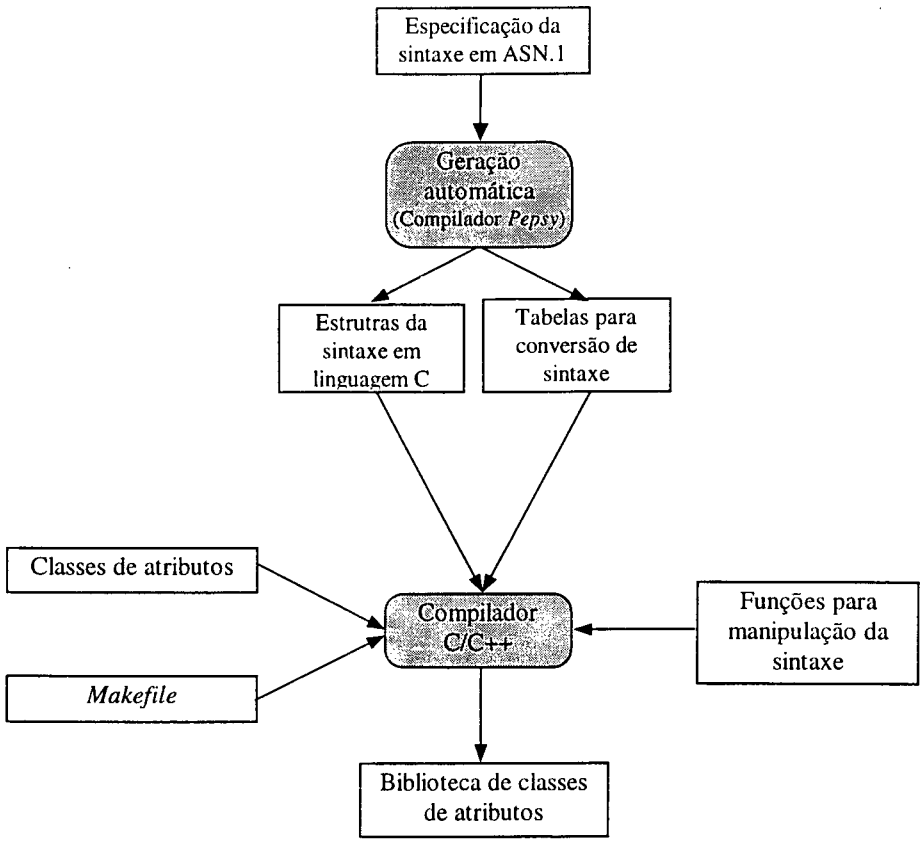

Figura 4.1: Diagrama da construção da biblioteca de classes de atributos.

Para cada sintaxe de tipo de atributo é então implementada uma classe a qual utiliza as estruturas geradas pelo compilador *Pepsy* e as funções para a manipulação de sintaxe de tipos de atributos desenvolvidas pelo projetista da

aplicação de gerência. Todos os códigos fonte gerados, inclusive aqueles produzidos pelo *Pepsy,* são processados pelo compilador C/C++. O processo de compilação resulta em uma biblioteca de atributos que posteriormente será utilizada na construção de objetos gerenciados.

4.1.4 Especificação de um objeto gerenciado em GDMO

Durante a implementação dos objetos gerenciados deve ser utilizado o compilador GDMO fornecido pela plataforma OSIMIS. Este compilador gera a estrutura básica de uma classe de objetos gerenciados a partir de uma especificação em GDMO. Parte das funções para manipulação dos componentes de um objeto como atributos, notificações e ações é gerada pelo compilador. Permanece mantida sob responsabilidade do projetista de aplicações de gerência, o desenvolvimento da sintaxe destes componentes e a implementação do seu comportamento.

Uma das vantagens da utilização do compilador GDMO está relacionada ao código C++ gerado, o qual mantém conformidade com o restante do código implementado na plataforma de gerenciamento. Esta conformidade facilita a integração com as interfaces para a construção de processos gerentes e agentes existentes no OSIMIS.

Com o objetivo de atender aos requisitos particulares de cada classe, além do código gerado pelo compilador, algumas funções devem ser implementadas pelo projetista da aplicação. Neste contexto estão inseridos os métodos e funções que implementam o comportamento dos atributos, ações e notificações.

Em acréscimo, devem ser implementadas as funções para a criação de identificadores (RDNs - *Relative Distinguished Names*) para os objetos instanciados.

Estes identificadores são necessários para viabilizarem o funcionamento do esquema de *name binding*. Este esquema é utilizado na construção da hierarquia de nomeação ou de *containment* a qual permite identificar, de forma unívoca, cada instância de objeto gerenciado no sistema.

4.1.5 Utilização do Compilador GDMO

A construção de classes de objetos gerenciados utilizando o compilador GDMO requer, além da especificação em GDMO, a elaboração de uma base de dados contendo informações sobre os objetos gerenciados, atributos e sintaxes:

- *mo.dat*: tabela dos objetos gerenciados referenciados na especificação GDMO. Cada entrada da tabela contém o nome de uma classe de objetos presente na especificação, o nome da classe C++ que a implementa e o nome do arquivo *header*[2] onde a classe está definida;
- *attr.dat*: todos os atributos utilizados na especificação são identificados nesta tabela. Cada entrada contém o nome de um atributo, a classe C++ que o implementa e o nome do arquivo *header* onde a classe está definida;
- *syntax.dat*: identifica as sintaxes empregadas na especificação. Cada entrada contém o nome de uma sintaxe, a classe C++ que a implementa e o nome do arquivo *header* onde a classe está definida.

[2] Em linguagens de programação um arquivo *header* é geralmente identificado pelo sufixo .h inserido no final do arquivo. O conteúdo de um arquivo *header* é restrito a declaração dos métodos que serão implementados em uma classe de objetos.

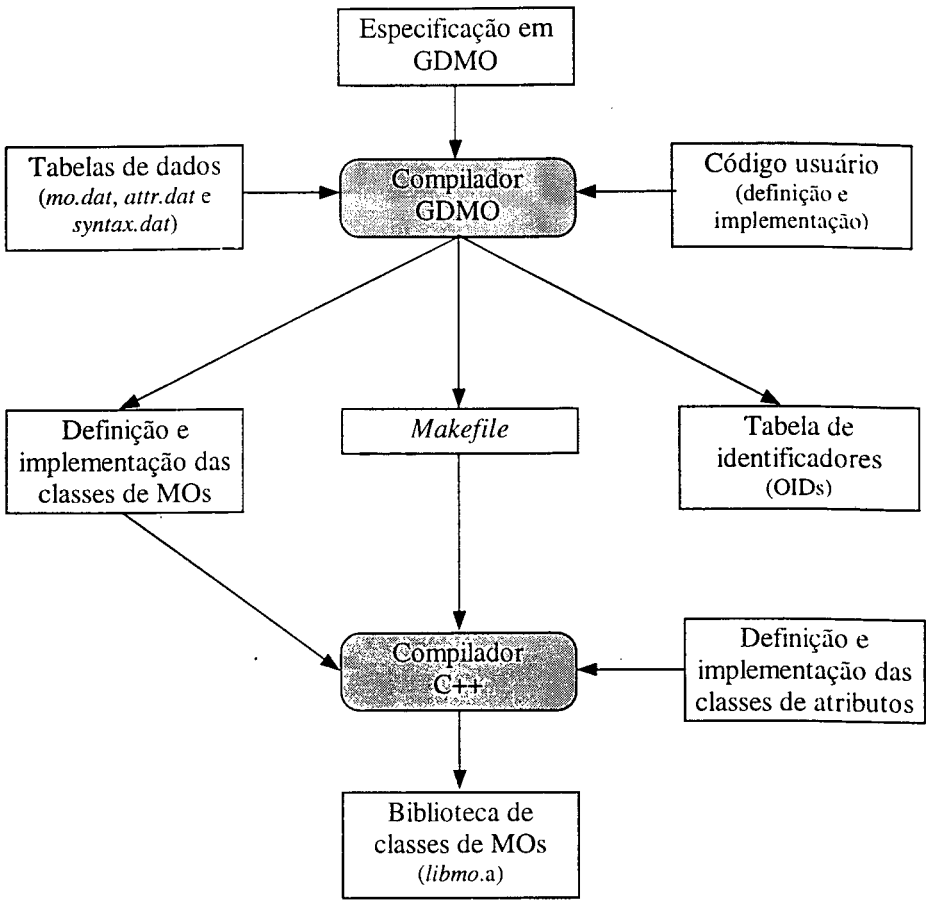

Figura 4.2: Desenvolvimento de objetos gerenciados utilizando o compilador GDMO.

Os arquivos com as tabelas e o arquivo contendo as especificações dos objetos gerenciados são enviados ao compilador GDMO, que gera os seguintes resultados:

- arquivos com a definição da estrutura básica das classes de objetos gerenciados especificados;
- arquivos com a implementação das classes especificadas;
- uma tabela contendo os identificadores de objetos[3] (OIDs) para as estruturas geradas; e,

[3] Na arquitetura de gerenciamento OSI , tanto objetos gerenciados, quanto pacotes, ações, notificações, atributos e *name bindings* recebem identificadores, chamados *object identifiers* (OIDs). Estes identificadores são obtidos a partir da hierarquia de registro.

- um arquivo *Makefile* com as diretivas de compilação para a construção de uma biblioteca com as classes definidas.

O diagrama da figura 4.2 representa os elementos envolvidos na construção da biblioteca com as classes de objetos gerenciados (*libmo.a*). A inclusão do código escrito pelo implementador é realizada por meio da criação de arquivos de definição (*header*) e de implementação. Estes arquivos são automaticamente incluídos pelo compilador GDMO, através da diretiva *#include*. Desta forma, o código do usuário permanece separado do código gerado automaticamente, garantindo que, a cada nova compilação realizada pelo compilador, o código implementado pelo projetista continue inalterado.

4.1.6 Inclusão dos aspectos semânticos das ações e notificações no código gerado pelo compilador

Após implementado os novos tipos de atributos, deverão ser construídas as ações as quais estarão submetidos os objetos gerenciados. A utilização de uma ação é opcional e sua presença depende da finalidade do objeto a ser implementado.

A descrição GDMO do comportamento de uma ação é realizada em linguagem natural e portanto, em muitos casos, passível de ambiguidades e erros de interpretação. Tais características são refletidas nas restrições do compilador GDMO as quais impossibilitam a geração automática do comportamento da ação para linguagem C++. Desta forma, a implementação dos aspectos semânticos da ação é mantida sob atividade do projetista da aplicação. Usualmente, a especificação do comportamento de uma ação contém somente características genéricas e sua

implementação deve ser, então, adaptada às particularidades do ambiente a qual está sendo desenvolvida.

As notificações, por sua vez, apresentam sintaxe similar aos atributos a qual identifica etapas de implementação correlatas. Os compiladores e as interfaces para a construção do argumento de uma notificação são idênticos aos observados durante a implementação da sintaxe de um atributo. Após a especificação em GDMO, os aspectos semânticos de uma notificação devem ser implementados pelo projetista da aplicação. Frequentemente as notificações apresentam um comportamento complexo em virtude de estarem dependentes , em muitos casos, da execução de demais operações sobre o objeto gerenciado. Como exemplo, uma notificação pode ser emitida após a realização de uma operação de criação ou remoção de um objeto, ou então, quando atendida uma condição que determine a emissão de relatório de eventos por parte de um objeto a outros processos de gerenciamento.

4.1.7 Integração do objeto gerenciado a um agente

Obtida a implementação dos requisitos básicos para os objetos gerenciados, a etapa subsequente consiste na integração destes objetos com a plataforma de gerenciamento.

A plataforma OSIMIS fornece um processo agente, denominado *System Management Agent* (SMA) o qual implementa um subconjunto das funções de gerenciamento de sistema definidas no modelo funcional da arquitetura OSI. Dentre as funções que constituem a estrutura do agente estão presentes as funções de gerenciamento de objeto, gerenciamento de estado, relatório de eventos, registro de *logs* e objetos métricos.

Extensões ao agente podem ser realizadas pela inclusão de novos objetos gerenciados em sua Base de Informação de Gerenciamento. Para tanto, tabelas de identificadores de objetos mantidas pelo agente devem alteradas com o objetivo de que sejam identificados os componentes dos objetos gerenciados recém implementados.

- **tabela de objetos gerenciados**: nesta tabela são listadas as classes de objetos que serão referenciadas após a inicialização da MIB;
- **tabela de notificações**: esta tabela identifica as notificações que poderão ser emitidas, realizando um mapeamento destas para as respectivas classes de objetos gerenciados;
- **tabela de sintaxes**: identifica todas as sintaxes (para ações, notificações e atributos) que serão suportadas pelo agente após sua inicialização. Cada entrada desta tabela contém o nome de uma rotina, responsável pela inicialização das sintaxes de uma função de gerenciamento.

Realizadas as devidas alterações no agente, este deve ser compilado novamente com o objetivo de incluir as bibliotecas *libmo* e *libsntx*, as quais correspondem respectivamente a implementação dos objetos gerenciados e das sintaxes dos atributos.

O modelo de gerenciamento OSI utiliza-se de um protótipo do serviço de diretório X.500 para identificar, de forma única, todas as entidades de um sistema. O OSIMIS implementa mecanismos de suporte a este serviço. No entanto, caso estes mecanismos não estejam instalados, pode ser utilizado um meio alternativo, que faz uso de tabelas para mapear os diversos identificadores de objetos. Desta forma, identificadores relacionados a atributos, ações e notificações estão presentes na

tabela denominada *oidtable.at*. Os identificadores relacionados aos objetos gerenciados e *name bindings* deverão ser incluídos na tabela denominada *oidtable.gen*.

A etapa seguinte consiste na inclusão dos novos identificadores de objetos (OIDs) nas tabelas de identificadores da plataforma OSIMIS. Por fim, a operação de criação de objetos é realizada a partir de um *script*[4], que invoca o gerente *mcreate*. Este gerente é fornecido pela plataforma OSIMIS e apresenta como parâmetros, entre outros, o nome do agente (SMA), o servidor da rede a qual está instalado agente, o nome da classe de objeto a ser instanciada e o identificador da instância superior sob o qual estará subordinado o objeto.

4.1.8 Utilização das interfaces SMIB e RMIB para a implementação de processos gerentes

Segundo [Pav93] os projetos associados ao desenvolvimento da plataforma OSIMIS têm atuado na construção de interfaces para o acesso ao protocolo CMIP. Estas interfaces fornecem eficiente acesso a MIBs remotas, facilitando desta forma a construção de processos gerentes. Construídas em conformidade com o paradigma de orientação a objetos, estas APIs de acesso ao protocolo CMIP foram implementadas no OSIMIS, por meio de duas abordagens distintas: *Remote MIB* e *Shadow MIB*.

A interface *Remote MIB* (RMIB) fornece uma abstração de MIBs remotas OSI empregando a noção de um objeto de associação. Este objeto atua como interface no estabelecimento de uma associação de gerenciamento com um agente remoto,

[4] *Scripts* são arquivos escritos em uma linguagem especial para o interpretador de comandos (*shell*) do sistema UNIX. A linguagem possibilita a declaração de variáveis e provê estruturas de controle (*if, while,...*), além de operações lógicas e aritméticas.

ocultando a utilização dos parâmetros do protocolo CMIP e os requisitos de acesso à árvore de informação de gerenciamento localizada em um nó remoto.

O conceito de *Shadow MIB* (SMIB) é apresentado pela abstração dos objetos em espaço de endereçamento local e possibilita que a utilização do protocolo de gerenciamento seja completamente ocultado. Parâmetros de acesso aos objetos como escopo e *Distinguished Names* podem ser substituídos por ponteiros.

Os procedimentos para a construção de processos gerentes independem dos requisitos para a construção de agentes possibilitando que ambos os processos sejam construídos separadamente. No entanto, esforços devem ser observados com o objetivo de manter conformidade com os meios de acesso ao protocolo CMIP de modo que seja garantida a comunicação entre os processos de gerenciamento.

5. Desenvolvimento de Extensões e Resultados

Uma tarefa constantemente adotada nas empresas de telecomunicações consiste no estabelecimento de tarifas para a utilização de recursos. Em algumas redes de computadores a utilização de recursos é monitorada não somente para o estabelecimento de tarifas, como também para o controle e a distribuição de cotas de consumo aos usuários [KoCo95a]. Para a realização destas atividades deve existir um mecanismo de contabilização que seja genérico o suficiente, de forma a possibilitar o processamento de dados de acordo com os requisitos dos sistemas onde a atividade de contabilização seja necessária. Desta forma, o mecanismo deve ser capaz de contabilizar o uso de equipamentos e serviços tão distintos como uma impressora ou uma central telefônica.

A gerência de contabilização surge para atender estes requisitos, possibilitando, além da monitoração da utilização dos recursos, a associação de taxas a serem cobradas por esta utilização. Em acréscimo fornece, juntamente com a gerência de desempenho, parâmetros que auxiliam os administradores na tomada de decisões a respeito da aplicação de novos recursos ou da redistribuição daqueles já existentes.

Um exemplo típico de aplicação da gerência de contabilização pode ser observado em redes de telecomunicações. Nestes ambientes, é fundamental a provisão de meios para a tarifação dos diversos serviços prestados aos seus usuários e assinantes, tais como, telefonia, fac-símile, transmissão de dados e outros. As tarifas de utilização podem não ser fixas, variando de acordo com as informações provindas da gerência contabilização. Desta forma, tarifas distintas

podem ser estabelecidas nos períodos em que há uma maior utilização dos recursos.

Mesmo em ambientes onde não se aplica tarifação sobre a utilização de recursos e serviços, a gerência de contabilização pode ser empregada como um meio auxiliar para garantir que os usuários façam uso adequado dos recursos, possibilitando que todos tenham acesso garantido aos recursos que realmente necessitam. Para tanto, cotas de utilização podem ser definidas de acordo com as prioridades específicas de cada grupo de usuários. A contabilização do uso dos recursos, possibilita que se verifique o cumprimento das cotas por parte dos usuários.

O modelo funcional da arquitetura de gerenciamento OSI especifica cinco áreas funcionais as quais englobam, entre outras, a gerência de contabilização. Cada área funcional define um conjunto de uma ou mais funções, cuja implementação provê os requisitos para o gerenciamento de sistemas. No caso da gerência de contabilização, estes requisitos são especificados através da Função de Medida de Uso, definida em [ISO 10164-10b].

Em virtude da relevância dos serviços de contabilização em redes de telecomunicações, a abordagem inicial consistiu na construção de aplicações de gerenciamento para recursos específicos da rede. No entanto, uma vez observada que a plataforma OSIMIS não fornecia facilidades relacionadas à gerência de contabilização, os objetivos foram alterados no intuito de concentrar esforços no enriquecimento da plataforma OSIMIS com uma estrutura de suporte ao desenvolvimento de aplicações de contabilização. Como a plataforma está em conformidade com o modelo de gerenciamento OSI, a implementação da Função de Medida de Uso representou a melhor abordagem.

5.1 Função de Medida de Uso [ISO 10164-10b]

A Função de Medida de Uso deve ser implementada seguindo as especificações contidas na recomendação [ISO 10164-10b]. Este documento define, entre outros aspectos, os serviços fornecidos pela função, o protocolo necessário para suportar esses serviços e o relacionamento do serviço de contabilização com as outras funções do modelo funcional OSI. Em acréscimo a estas especificações, a recomendação define, através de especificações em GDMO, as classes de objetos gerenciados, os pacotes, os atributos, as operações e as notificações necessários para a implementação da função.

5.1.1 Requisitos para a contabilização de recursos

A Função de Medida de Uso deve prover meios para medida e coleta de informações a respeito da utilização de recursos e serviços na rede. Em acréscimo, a obtenção e a representação das informações de contabilização devem ser padronizadas, de modo a garantir a interoperabilidade dos serviços que seguem o modelo OSI. A função deve habilitar os sistemas de gerenciamento a coletar e controlar a obtenção de dados relativos ao uso dos recursos em um ambiente OSI. Deve também, garantir que estes dados estejam disponíveis quando requisitados pelo sistema de gerenciamento, tanto durante o processo de coleta, quanto num momento posterior.

Existem várias formas pelas quais pode ser realizada a coleta dos dados. No entanto, a função de contabilização deve ser genérica o suficiente para possibilitar que os dados coletados sejam tratados adequadamente, de acordo com as necessidades de cada aplicação. Em determinadas situações, os dados coletados podem ser utilizados para a aplicação de tarifas sobre os serviços; em outros casos,

estes dados podem fornecer meios para o controle e a distribuição de cotas de consumo aos usuários.

Além de incluir aspectos relacionados a coleta e ao controle da atividade de medida, a Função de Medida de Uso especifica objetos de registro que atuam no armazenamento dos dados referentes a utilização de determinado recurso, desta forma, tornar tais dados disponíveis para posterior consulta.

5.1.2 Modelo para a Medida de Contabilização

A medida de contabilização é modelada através dos objetos gerenciados associados à contabilização da utilização dos recursos. Estes objetos por sua vez, estão associados aos chamados "objetos contabilizados", que representam as demais características dos recursos monitorados.

Dois aspectos devem ser levados em consideração na implementação da Função de Medida de Uso:

- o primeiro aspecto diz respeito ao controle do registro e emissão dos dados associados com a contabilização, sendo representado por objetos gerenciados denominados **objetos de controle de medida de uso** (*usage metering control objects*);
- o segundo, refere-se especificamente aos dados coletados, sendo representado pelos **objetos de dados de medida de uso** (*usage metering data objects*).

5.1.2.1 Objetos de Controle de Medida de Uso

Estes objetos habilitam o sistema de gerenciamento a coletar informações sobre a utilização de um recurso, selecionando quais os dados relevantes e sob quais circunstâncias esta coleta deve ser realizada.

O controle de medida especifica os eventos que resultam na atualização e notificação das informações sobre a utilização dos recursos. Tais eventos incluem o escalonamento por períodos de tempo, ações de controle do próprio sistema de gerenciamento e estímulos externos.

Os objetos de controle de medida fornecem uma visão genérica do gerenciamento, podendo ser especializados para a contabilização de recursos específicos. Outras classes de objetos gerenciados podem fazer uso dos pacotes definidos no controle de medida para incorporarem funcionalidades inerentes à contabilização.

5.1.2.1.1 Pacotes para o Controle de Medida de Contabilização

Os objetos de controle de medida de contabilização são definidos a partir de pacotes, que agrupam atributos, ações e notificações. Tais pacotes são descritos a seguir:

Metering Control Object

Este pacote é obrigatório e especifica dois atributos:

- *control object id*: é um identificador para o objeto de controle de medida; e
- *operational state*: indica o estado operacional do objeto, podendo assumir os valores *ENABLED* ou *DISABLED*.

Metering Control Capabilities

Pacote obrigatório que inclui os seguintes atributos:

- *reporting triggers*: especifica os eventos[5] que resultarão na atualização dos dados de medida de contabilização;
- *accountable objects reference list*: é um conjunto de referências aos objetos contabilizados sobre os quais o objeto de controle de medida atua; e,
- *data object reference list*: é um conjunto de referências aos objetos de dados de medida (*usage metering data objects*) controlados pelo objeto de controle.

Metering Start

Pacote condicional presente caso os objetos de dados de medida sejam criados como objetos inativos, isto é, em um estado de bloqueio (*suspended*), podendo ser ativados posteriormente. Contém uma única ação:

- *start metering*: esta ação permite que um gerente inicie ou retome as medidas de contabilização. Os objetos de dados de medida especificados na operação saem do estado *suspended* e iniciam suas atividades de monitoração.

Metering Control

Este pacote está presente quando forem solicitadas operações para suspender ou retomar as medidas de contabilização. Este pacote provê duas ações:

[5] Os eventos que causam a atualização ou informação dos dados de contabilização podem ser classificados em três formas:
- Escalonamento de intervalos periódicos;
- Resultado de ações que identificam o início, a suspensão ou a retomada das medidas;
- Estímulo decorrente da alteração do valor de um atributo.

- *suspend metering*: permite que um gerente interrompa as atividades de medida para os objetos selecionados na operação. O valor *suspended* é adicionado ao *status* desses objetos. Os dados já coletados são mantidos inalterados para uma posterior retomada das atividades; e,
- *resume metering*: esta ação é utilizada pelo gerente para retomar as atividades de medida de contabilização. O valor *suspended* é retirado do *status* dos objetos gerenciados selecionados na operação.

Start Notification

Contém uma única notificação:

- *metering started*: informa um gerente (diferente daquele que solicitou a operação) sobre o resultado da execução de uma ação *start metering.*

Control Notification

Define duas notificações:

- *metering suspended*; e
- *metering resumed.*

De maneira análoga à ação do pacote anterior, estas notificações informam um gerente sobre o resultado da execução de uma ação *suspend metering* ou *resume metering.*

Os três tipos de notificações definidos anteriormente, são gerados como conseqüência da conclusão das ações aplicadas sobre os objetos de controle. As informações contidas nessas notificações identificam os objetos de dados sobre os quais a ação foi aplicada e os valores de alguns dos atributos do objeto de controle.

5.1.2.2 Objetos de Dados de Medida de Uso

O objeto de dado de medida representa a utilização de um recurso por parte de um usuário, contendo informações que identificam, entre outras, o usuário do recurso, a unidade de medida utilizada e a quantidade consumida.

As informações da utilização de um recurso podem ser obtidas por meio de duas abordagens: pelo uso da operação GET, para obter valores de atributos de dados de medida; ou pela inclusão de parâmetros de medida nas notificações de controle de contabilização.

Apenas as propriedades genéricas dos objetos de dados de medida são definidas pelo padrão. Especializações destes objetos podem ser elaboradas para a contabilização de recursos específicos.

5.1.2.2.1 Pacotes para Dados de Medida de Contabilização

Os objetos de dados de medida de contabilização são definidos a partir de pacotes, que agrupam atributos e notificações, com seus respectivos comportamentos. Tais pacotes são descritos a seguir:

Metering Data Object

É um pacote obrigatório, que contém um único atributo:

- *data object id*: especifica um identificador para o objeto de dados de medida.

Metering Data Info

Agrupa os atributos que efetivamente representam os dados sobre a utilização de um recurso por um usuário ou assinante:

- *usage information:* fornece a informação de medida de uso relativa a qualquer tipo de recurso ou serviço;
- *accountable object reference:* identifica o nome de uma instância de um objeto gerenciado;
- *data errors:* utilizado para indicar possíveis erros associados aos dados coletados. Pode a assumir os valores *POSSIBLE ERROS* e *NO ERROS*. O primeiro identifica um objeto que representa possível causa do erro, enquanto o segundo indica a inexistência do erro; e
- *provider id:* identifica a fornecedor do serviço prestado.

Metering Data Condition

Contém atributos que definem a condição do objeto de dados no que se refere às atividades de medida de utilização:

- *control status*: assume o valor *suspended* quando o objeto de dados não está contabilizando o uso do recurso; e
- *procedural status*: assume o valor *terminating* quando o objeto de dados interrompe a coleta de medidas como resultado de uma operação *delete*.

Audit Information

Este pacote deve ser incluído caso seja requerido suporte a facilidades de auditoria na atividade de medida de uso. Este pacote contém um único atributo:

- *audit information:* apresenta informações sobre a origem da qual deriva o dado de medida de uso.

5.1.3 O Relacionamento entre Objetos Contabilizados, Objetos de Controle de Medida e Objetos de Dados de Medida

Tanto o controle de medida quanto os dados de medida de uso podem ser modelados como objetos gerenciados distintos, ou como partes dos objetos gerenciados que representam os recursos a serem contabilizados.

De acordo com [ISO 10164-10b], os objetos de controle de medida de uso fornecem referências aos objetos de dados de medida os quais controlam. Cada instância de objeto de controle pode controlar várias instâncias de objetos de dados. Cada objeto de dados de medida mantém uma referência à instância de objeto contabilizado sobre o qual ele realiza a coleta de informações.

É importante observar também que os objetos de controle e objetos de dados de medida podem estar presentes no mesmo contexto em que está o objeto contabilizado, ou podem estar contidos no próprio objeto contabilizado.

5.1.4 Considerações sobre a operação com os Objetos de Contabilização

Um objeto contabilizado somente estará disponível a um usuário que deseja realizar atividades de monitoração de sua utilização quando houver um objeto de controle de medida associado.

Para que uma solicitação sobre um objeto contabilizado seja concedida é necessário que exista pelo menos uma instância de objeto de dado de medida responsável pela sua monitoração.

Um objeto de dados de medida só pode ser criado caso exista uma instância de objeto de controle de medida para controlá-lo. Em contrapartida, um objeto de controle de medida somente poderá ser removido a partir do momento que todos os objetos de dados sob o seu controle tenham sido removidos.

Após a criação dos objetos de dados de medidas, o objeto de controle pode executar solicitações para iniciar, suspender e retomar as medidas de contabilização.

5.2 Implementação de Serviços de Contabilização

A Função de Medida de Uso tem por objetivo permitir a coleta e o controle da coleta de dados sobre a utilização dos recursos e serviços disponíveis na rede. Objetiva também, criar registros sobre os dados coletados, permitindo seu posterior processamento. Os dados registrados podem ser processados, por exemplo, para se realizar a associação de tarifas com a utilização dos recursos.

Para a obtenção da funcionalidade de coleta, controle e registro de dados sobre a utilização de recursos, a normalização [ISO 10164-10b] proposta no modelo funcional OSI, sugere a implementação de três classes de objetos gerenciados:

- *usage metering control object*: os objetos desta classe são responsáveis por permitir o controle sobre o início, suspensão e retomada das medidas de contabilização mantendo informações sobre como devem ser realizadas estas medidas;
- *usage metering data object*: seus objetos realizam efetivamente a coleta de dados sobre a utilização de um recurso por um usuário ou assinante. Estes objetos são controlados pelos objetos de controle e guardam informações relativas aos objetos que representam os recursos contabilizados;
- *usage metering record object*: os objetos desta classe são requisitados para compor um histórico dos dados coletados pelos objetos de dados de medida.

A recomendação da Função de Medida de Uso fornece estruturas em GDMO para as três classes de objetos gerenciados. Estas estruturas incluem a especificação completa dos atributos, ações, notificações e comportamentos que compõem os objetos gerenciados.

As seções subsequentes apresentam os detalhes da implementação da Função de Medida de Uso para a plataforma OSIMIS/ISODE. São abordados aspectos sobre a implementação de atributos, ações e notificações, até a obtenção das novas classes de objetos gerenciados. Em acréscimo, são descritos os processos para a criação de *name bindings*, que definem o relacionamento de nomeação entre as classes de objetos. Finalmente, são abordadas as questões referentes à integração da função implementada com a plataforma de gerenciamento.

5.3 Atributos

Apesar da plataforma OSIMIS fornecer implementações das classes dos atributos comumente utilizados, os objetos definidos na Função de Medida de Uso são compostos por uma série de atributos inexistentes na plataforma. Foi evidenciada então a necessidade da construção de 15 novas classes de atributos, como pode ser observado na tabela I. Cada linha desta tabela identifica a sintaxe do atributo especificada na Função de Medida de Uso , a sua respectiva estrutura de dados em linguagem C, a classe C++ que a implementa e a classificação do atributo sob o aspecto de ser multi-valorado.

Tabela I: Lista dos atributos implementados.

Sintaxe do atributo	Estrutura em C	Classe em C++
AccountableObject-Reference	type_UMFObjId_Accountable-ObjectReference *	AccountableObject-Reference
AccountableObject-ReferenceList	type_UMFObjId_Accountable-ObjectReferenceList *	AccountableObject-ReferenceList
ActionArgument	type_UMFObjId_ActionArgument *	ActionArgument
ActionResponse	type_UMFObjId_ActionResponse *	ActionResponse
AuditInfo	type_UMFObjId_AuditInfo *	AuditInfo
ControlInfo	type_UMFObjId_ControlInfo *	ControlInfo
ControlStatusValue	type_UMFObjId_ControlStatusValue*	ControlStatusValue
DataErrors	type_UMFObjId_DataErrors *	DataErrors
DeniedMeteringAction	type_UMFObjId_ DeniedMeteringAction *	DeniedMeteringAction
NotificationCause	type_UMFObjId_NotificationCause *	NotificationCause
ProceduralStatusValue	type_UMFObjId_ProceduralStatusValue *	ProceduralStatusValue
ReportingEvent	type_UMFObjId_ReportingEvent *	ReportingEvent
ReportingTriggers	type_UMFObjId_ReportingTriggers *	ReportingTriggers
UsageDataInfo	type_UMFObjId_UsageDataInfo*	UsageDataInfo
UsageInfo	type_UMFObjId_UsageInfo*	UsageInfo

Como observado no capítulo 4, a implementação de uma classe de atributo pode ser dividida nas seguintes etapas: especificação ASN.1 da sintaxe dos atributos (vide anexo 11.1), utilização do compilador *Pepsy* para a geração automática de estruturas em linguagem C, desenvolvimento de funções para manipulação de sintaxe (vide anexo 11.4) e, finalmente, a construção da classe para um novo tipo de atributo (vide anexo 11.3).

5.4 Ações

A Função de Medida de Uso é composta por três classes de objetos gerenciados, que realizam o controle das medidas (*usage metering control objects*), a coleta das informações sobre a utilização dos recursos (*usage metering data*

objects), e o registro destas informações (*usage metering records*). Dentre estas, somente para a classe de objetos de controle de medida estão associadas ações.

Estas ações provêm meios para que um agente possa controlar, em resposta às solicitações de um gerente, a coleta de dados sobre a utilização dos recursos, através de operações sobre os objetos de controle. Tais ações possibilitam que se comande o início (*start metering*), a suspensão (*suspend metering*) e a retomada das medidas (*resume metering*).

Cada objeto de controle atua sobre um ou mais objetos de dados de medida. Quando um agente realiza uma operação sobre um objeto de controie para determinar o início, a suspensão ou a retomada das medidas de contabilização, ele deve especificar quais objetos de dados devem ser monitorados.

Quando o objeto de controle recebe uma operação do agente é alterado o *status* de cada objeto de dados selecionado. Se a operação for para iniciar ou retomar a coleta de medidas, o valor *suspended* é retirado do atributo de *status* dos objetos de dados, fazendo com que estes reiniciem as medidas. No caso de uma operação para suspensão da coleta, o atributo de *status* dos objetos de dados recebe o valor *suspended*.

As ações *startMetering*, *suspendMetering* e *resumeMetering*, possuem um parâmetro que identifica os objetos de dados de medida sobre os quais as ações terão efeito. Este parâmetro é representado, em cada operação, por uma instância da classe de atributo *ActionArgument*, cuja sintaxe pode ser visualizada na figura 5.1. O parâmetro pode assumir dois valores: uma lista de identificadores para os objetos de dados de medida selecionados; ou um valor nulo, indicando que todos os

objetos de dados subordinados devem ser afetados. Tal estrutura pode ser observada na figura 5.1.

```
ActionArgument ::=
   CHOICE {
      selectedObjects     ObjectInstanceList,
            -- set of data objects, controlled by the control
            -- object for which the request is appropriete
      allObjects          NULL
            -- selects all data objects controlled by control
            -- object
         }
```

Figura 5.1: Especificação ASN.1 da sintaxe da classe de atributos ActionArgument.

As ações possuem também um argumento de retorno, que informa sobre o resultado de cada operação realizada. Este argumento é representado por um atributo da classe *ActionResponse*, indicando os objetos de dados sobre os quais a ação foi realizada com sucesso, aqueles sobre os quais a ação falhou e aqueles cujo resultado da ação não pode ser determinado. A sintaxe para este tipo de atributo pode ser observada na figura 5.2.

```
ActionResponse ::=
    SEQUENCE {
                  -- at least one component shall be present
      success             [0] ObjectInstanceList    OPTIONAL,
      failed              [1] ObjectInstanceList    OPTIONAL,
      indeterminate       [2] ObjectInstanceList    OPTIONAL

               }
```

Figura 5.2: Especificação ASN.1 da sintaxe da classe de atributos ActionResponse.

Cada ação possui um comportamento próprio que é implementado no método *action* da classe de objetos de controle. Em resposta a uma solicitação de um gerente, este método é invocado pelo agente para realizar uma ação sobre um objeto gerenciado.

5.5 Notificações

O serviço de contabilização implementado define as notificações necessárias para informar os processos gerentes sobre os eventos ocorridos com os objetos gerenciados de controle e dados de medida.

Para a classe de objetos de controle são definidas três notificações: *metering started*, *metering suspended* e *metering resumed*. Estas notificações são emitidas por um objeto de controle para informar um gerente sobre a realização de uma ação de controle que tenha determinado o início, a suspensão ou a retomada das medidas de contabilização.

As notificações anteriormente citadas possuem um argumento representado pela estrutura de dados denominado *control info*. Este argumento possui informações sobre o resultado da ação que acarretou a emissão da notificação. Em acréscimo, sua estrutura contém dados sobre o objeto de controle que está emitindo a notificação, tais como, o resultado de sucesso ou fracasso na realização da ação e o motivo pelo qual foi emitida a notificação.

Para as notificações não foi necessária a implementação de classes. Somente as funções para possibilitar a codificação e decodificação das estruturas entre as linguagem C e ASN.1 foram escritas.

No caso da ciasse de objetos de dados de medida, foi definida uma única notificação que é emitida a um gerente quando se deseja informar sobre a coleta de novos dados de utilização de um recurso por um usuário ou assinante. O argumento associado a esta notificação é representado pela estrutura denominada *usage data info* que apresenta uma série de informações a respeito do objeto de dados, tais

como, o identificador do objeto contabilizado, a unidade de medida utilizada, a quantidade despendida e o tipo de tarifa associada.

5.6 Objetos Gerenciados

No caso da Função de Medida de Uso, as classe de objetos (*usageMeteringControl*, *usageMeteringData* e *usageMeteringRecord*) foram parcialmente implementadas pelo compilador GDMO (vide anexo 11.2). Além do código gerado pelo compilador, métodos para a manipulação de ações e notificações foram implementados manualmente, para atender aos requisitos particulares de cada classe. No caso da classe *usageMeteringControl*, por exemplo, o método *action*, presente na interface para a construção de processos agente, foi especializado para implementar protótipos das ações inerentes à classe.

Em acréscimo, foram implementadas funções para a geração de identificadores (RDNs - *Relative Distinguished Names*) para os objetos instanciados. Estes identificadores são necessários para viabilizarem o funcionamento do esquema de *name binding*. Este esquema é utilizado na construção da hierarquia de nomeação ou de *containment* que permite identificar, de forma exclusiva, cada instância de objeto gerenciado no sistema.

O regime de *Polling* foi utilizado com o objetivo de implementar a interface com os recursos reais e testar a coleta de dados referentes à contabilização. A atividade de *Polling* é iniciada, suspensa ou retomada utilizando métodos herdados do mecanismo de coordenação de eventos implementado na plataforma OSIMIS. Também são implementados no OSIMIS métodos para monitoração de portas de comunicação e, desta forma, permitir a análise dos dados no momento em que estes trafegam em tais portas. Esta abordagem assíncrona reduz o tráfego de

informações de gerenciamento em situações nas quais o ponto de comunicação externo é mantido ocioso por períodos de tempo relativamente longos [CoRo95] [CoRo96].

A última etapa na implementação dos novos requisitos envolveu a integração da Função de Medida de Uso com a plataforma OSIMIS. Para tanto, foram realizadas alterações no agente SMA (*System Management Agent*) de forma a tornar os objetos implementados acessíveis aos processos gerentes. O agente SMA implementa um subconjunto das funções propostas no modelo funcional OSI. Finalmente foram elaborados *script* para a criação e remoção de instâncias dos objetos gerenciados anteriormente descritos, bem como, *scripts* para a execução das ações definidas para o processo de medida de uso.

6. Principais Conceitos sobre o Modelo TMN

O modelo TMN (*Telecommunication Management Network*) como proposto pelo ITU-T, antigo CCITT, foi introduzido na metade da década de 80 e atualmente é amplamente adotado como padrão para a gerência de ambientes de telecomunicações e constitui uma solução viável para os requisitos encontrados em redes de alta velocidade.

O escopo de uma rede TMN pode envolver diversos ambientes e atuar no gerenciamento de redes telefônicas, redes locais, redes de longa distância e inclusive a própria rede TMN. Desta forma, sob o domínio de uma rede TMN podem ser gerenciados uma infinidade de equipamentos de transmissão, todos interconectados por diferentes meios de transmissão.

O projeto de uma arquitetura TMN considera três aspectos para o gerenciamento de uma rede de telecomunicações. Estes aspectos constituem as arquiteturas física, funcional e informacional do modelo TMN.

6.1 Arquitetura física

Os componentes da arquitetura física são representados por interfaces e blocos de construção. Os blocos de construção correspondem aos diferentes tipos de componentes físicos de uma TMN, enquanto que as interfaces definem a troca de informação entre estes blocos. Um exemplo da arquitetura física TMN pode ser observada na figura 6.1 a qual reflete uma ilustração presente em [BRISA].

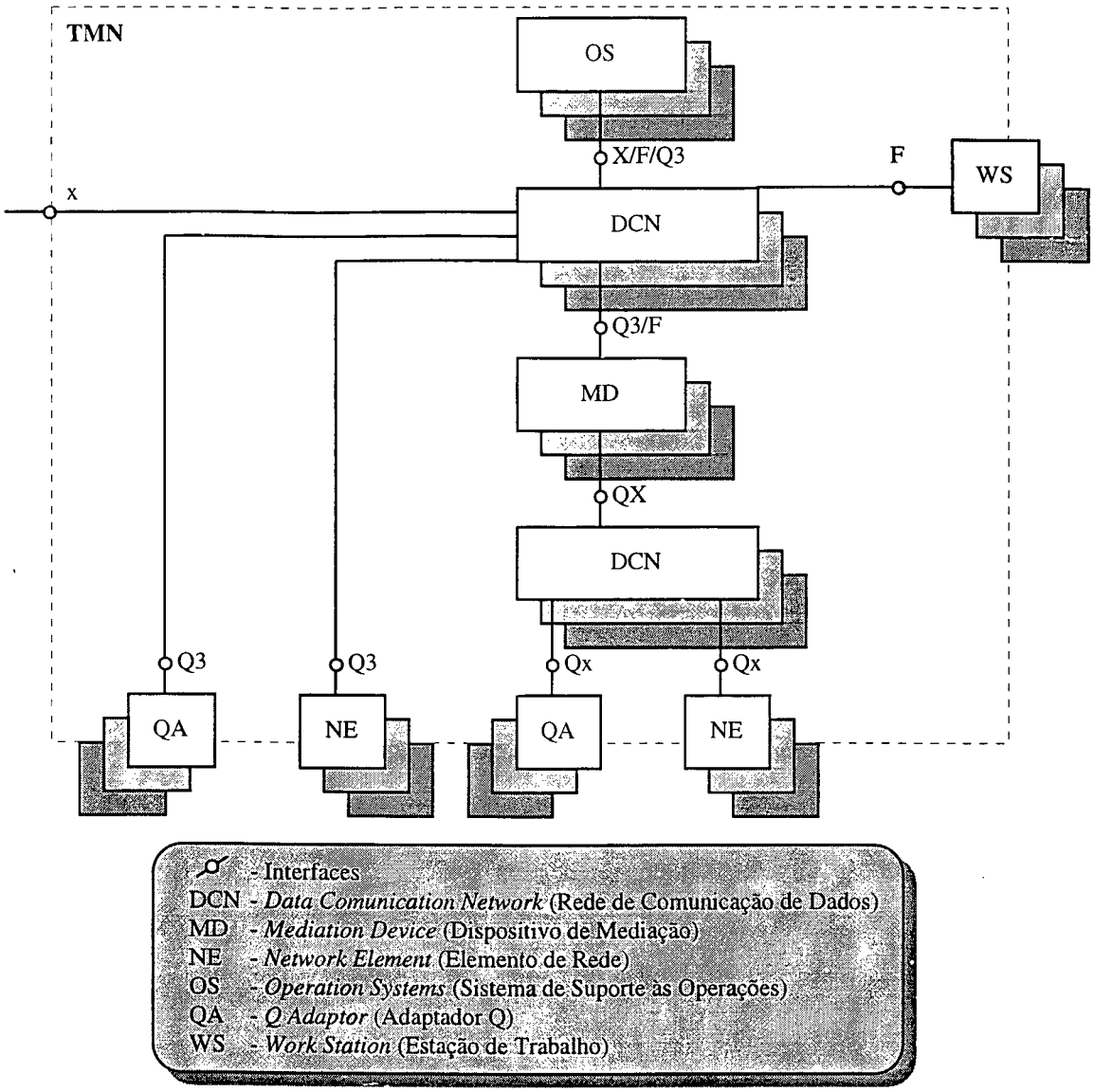

Figura 6.1: A arquitetura física TMN [BRISA].

O bloco Sistema de Suporte a Operações (*Operations System* - OS) atua no papel de gerente, realizando o processamento da informação relacionada ao gerenciamento de telecomunicações, com o objetivo de monitoração e controle dos demais componentes TMN.

Os Elementos de Rede (*Network Elements* - NE) implementam as funções que servirão de suporte para as atividades controle exercidas pelo Sistema de Operação. Estas funções de suporte correspondem a parte do comportamento de

uma aplicação de gerência e podem atuar nas áreas de configuração, contabilização, falhas, segurança e performance.

Uma Rede de Comunicação de Dados (*Data Communication Network* - DCN) é responsável pelo estabelecimento de uma conexões físicas providas por enlaces formados a partir de diferentes tipos de componentes de rede.

Adaptador-Q (*Q-Adaptor* - QA) é utilizado para estabelecer uma interface entre componentes TMN com entidades não TMN. O emprego de adaptadores-Q torna possível a utilização de conceitos TMN para o gerenciamento de uma ampla variedade de recursos que adotam uma arquitetura de gerenciamento proprietária ou seguem outros padrões de gerenciamento.

Dispositivos de Mediação (*Mediation Devices* - MD) são utilizados para a troca de informações de um Sistema de Operação para um Elemento de Rede ou um Adaptador-Q. Os Dispositivos de Mediação também podem estar associados às Estações de Trabalho (*Workstations* - WS). Estes últimos fornecem os meios para converter informações TMN para um formato apropriado ao usuário, e vice-versa.

Por fim, pontos de referência representando a troca de informações entre dois blocos de gerenciamento são representados pelas interfaces q, f, e X.

6.2 Arquitetura Funcional

A arquitetura funcional contém os blocos funcionais que implementam os serviços oferecidos pela arquitetura física. Esta arquitetura caracteriza a atividade de gerenciamento em cinco camadas:

- Camada de Elemento de Rede - corresponde aos componentes passíveis de gerenciamento existentes em rede de telecomunicações.

- Camada de Gerenciamento de Elemento de Rede - fornece uma visão particular de cada recurso existente na rede. Parte da atividade realizada por esta camada corresponde à coleta das informações que serão enviadas sistema de operação.

- Camada de Gerenciamento de Rede - responsável pelo gerenciamento de todos os componentes de uma rede e atua como suporte para os serviços que requerem uma visão total da rede.

- Camada de Gerenciamento de Serviço - responsável pela manutenção dos serviços fornecidos aos usuários. Esta camada fornece também uma interface ao usuário e interage com outros serviços e a Camada de Gerenciamento de Rede.

- Camada de Gerenciamento de Negócio - raramente implementada em ambientes atuais de telecomunicações, esta camada representa o mais alto nível de abstração de uma rede e incorpora funções de planejamento e análises gerenciais.

6.3 Arquitetura de informação

A arquiterura de informação TMN utiliza os conceitos de orientação a objetos possibilitando desta forma a utilização dos princípios do gerenciamento OSI. Deste modo, a arquitetura TMN também faz uso dos conceitos de objeto gerenciado, agente e gerente. A troca de informações entre os processos gerentes e agentes é mantida sob a responsabilidade do serviço e protocolo CMIS/CMIP.

7. Tecnologia TMN para gerência de redes ATM

Um relevante aspecto a ser considerado consiste na situação atual em que parte dos recursos ATM existentes apresentarem uma interface de gerenciamento orientada para o modelo SNMP. Tal característica de gerenciamento SNMP inerente a grande parte dos recursos ATM existentes no mercado implica no uso de interfaces para a comunicação com elementos TMN.

A utilização de soluções TMN para o gerenciamento de redes defronta com as restrições do modelo SNMP e conseqüentemente com as diferentes abordagens utilizadas por ambos os modelos em relação à visão de um elemento de rede a ser gerenciado. Enquanto o modelo SNMP está apenas estruturado em tipos de objetos e apresenta um modelo de informação que abrange um recurso único da rede, o modelo TMN adota o conceito integral de orientação a objetos e fornece uma abstração de um ambiente de rede que pode se estender do meio físico até o nível de planejamento de uma empresa de telecomunicações.

As sub-seções a seguir fazem uma análise comparativa entre a aborgadem proposta pelo modelo TMN para o gerenciamento de configuração em arquiteturas de redes de alta velocidade e os recursos fornecidos pela base instalada de equipamentos ATM dotados de modelos de informação SNMP.

7.1 Descrição de aspectos de gerenciamento de conexões

O componente de rede sob análise neste trabalho corresponde as entidades envolvidas na transferência de informação através de uma camada de rede. Neste contexto, uma rede pode ser modelada por meio de componentes recursivos de acordo com os conceitos de **partição** e **camada**. O primeiro identifica os limites de

domínio contidos em uma rede e o segundo especifica a associação cliente servidor entre camadas de rede adjacentes.

Funcionalidades distintas são observadas em uma rede de transporte as quais envolvem aspectos relacionados ao processamento da informação e à associação entre componentes de rede. A transferência de informação entre pontos terminais de uma camada de rede é mantida sobre a responsabilidade de uma entidade de transporte denominada *trail*. Em termos de conectividade, dois aspectos distintos devem ser considerados, a conexão contida no domínio de uma única sub rede a qual é denominada **conexão de sub-rede**, e a transferência de informação entre duas sub-redes conhecida como **conexão de enlace.** A concatenação das conexões de enlace e sub-rede suportados por um *trail* formam um **enlace topológico** [G.803].

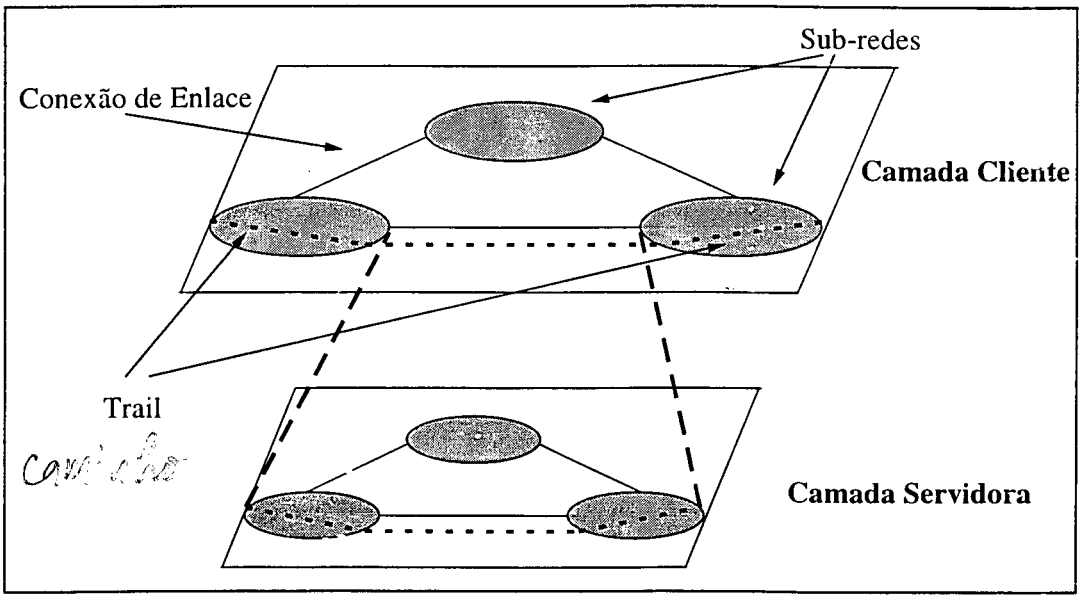

Figura 7.1: Principais componentes de uma partição de rede.

A figura 7.1 ilustra os principais componentes de uma partição de rede e identifica as entidades de transportes que serão responsáveis pela transmissão de informação. A camada de rede é composta por uma coleção de sub-redes que, em

uma abordagem recursiva, podem ser divididas em sub-redes menores e suas respectivas interconexões. Em um relacionamento cliente/servidor, conexões de enlace são fornecidos por *trails* contidos em uma camada de rede subjacente. Componentes adicionais como funções de terminação e adaptação estão localizados nos pontos de encontro de conexões e *trails*.

Em acréscimo as entidades básicas envolvidas em uma rede e as associações entre cada componente topológico, relacionamentos distintos podem ser encontrados quando a camada de rede é parte de diferentes domínios de gerenciamento. Estes relacionamentos apresentam definições opostas e representam a separação hierárquica e horizontal da camada de rede. A primeira corresponde ao relacionamento entre os diferentes níveis de uma rede e suas sub-redes contidas, enquanto que a última representa a divisão horizontal de uma rede separada por sub-redes interconectadas por meio de uma conexão de enlace.

7.1.1 Principais componentes de um fragmento de rede

Após identificados os componentes presentes em uma partição de rede, apresentadas as entidades envolvidas na transmissão da informação e verificado o relacionamento entre as principais entidades de transporte, será enfatizada a modelagem dos três componentes de transporte conhecidos como **rede**, **camada de rede** e **sub-rede**. As subseções a seguir fornecem uma análise sobre as diferentes abordagens especificadas nos atuais modelos de informação TMN e SNMP. Tal análise objetiva principalmente demonstrar a modelagem das entidades presentes no fragmento de rede.

Entretanto uma diferença significativa entre a abordagem adotada por estes dois modelos de informação deve ser inicialmente introduzida. Em termos de área

de aplicação, o modelo de informação TMN foi projetado para redes B-ISDN (*Broadband Integrated Services Digital Network*) orientadas a conexão e é provido com estruturas tecnológicas independentes que podem ser aplicadas em diferentes arquiteturas de rede, como no caso das arquiteturas ATM, SDH e SONET. Em contrapartida, o modelo de informação SNMP é especificamente direcionado a arquitetura ATM e desta forma descreve os equipamentos ATM em um nível de abstração inferior as estruturas presentes no modelo TMN.

7.1.2 Camada de Rede

A camada de rede é caracterizada pelo tipo de informação a ser transferida e é definida pelos pontos de acesso envolvidos na correspondente operação de transmissão [G.803]. Camadas de rede distintas contidas em uma rede manipulam diferentes tipos de informações compostas de sinais a taxas específicas. Os pontos de acesso a uma camada de rede são representados pelos pontos de terminação de *trail*.

O fornecimento e o uso de uma camada de rede caracteriza um relacionamento cliente/servidor o qual é suportado pelo modelo de informação TMN. Tal relacionamento significa que uma conexão na camada cliente deve ser estruturada sobre *trails* localizados na camada servidora. Como conseqüência, objetos específicos foram propostos no modelo TMN com extensões aos papéis de cliente e servidor. O fornecimento de um relacionamento cliente/servidor permite que instâncias de um único objeto de camada de rede possa atuar em papéis distintos de acordo com o objetivo corrente da camada de rede em prover suporte para a conexão das camadas superiores, ou em usar a camada de rede subordinada para o estabelecimento de conexões entre sub-redes contidas em uma

camada. Esta funcionalidade no modelo TMN é representada pela composição recursiva de sub-redes.

Uma abordagem distinta é observada no modelo SNMP para o gerenciamento de redes ATM. Neste caso, tipos de objetos são basicamente direcionados para o fornecimento de interfaces para os recursos físicos ATM. A metodologia adotada pelo modelo SNMP considera uma rede como um único comutador distribuído no qual toda a atividade de conexão é tratada localmente em cada recurso ATM. Como consequência, a principal contribuição consiste na modelagem e interfaceamento dos recursos físicos. Fazendo uma analogia com a arquitetura TMN, tal característica retrata o modelo de informação SNMP como mais compatível com os requisitos da camada de elemento de rede. Restrições são encontradas em termos da visão de gerenciamento da rede em virtude do fato de que o modelo SNMP não oferece suporte ao conceito de partição e camada o que resulta na subsequente impossibilidade de separar a rede em domínios hierárquicos e horizontais.

7.1.3 Rede

De acordo com [G.803] uma rede é definida como uma coleção de todos os recursos físicos e lógicos utilizados para fornecer serviços de telecomunicações.

Em contrapartida, o modelo SNMP, como mencionado anteriormente, manipula cada elemento de rede separadamente o que obriga o conjunto de informação de gerenciamento estar em constante interface com o recurso real subjacente. Não é possivel obter uma visão hierarquica da rede que, por questões de gerenciamento, poderia ser particionada em sub-redes menores. Ao contrário, para cada componente ATM devem ser fornecidas interfaces comuns sob o controle direto de sistemas de gerenciamento. Desta forma, informações de gerenciamento

são mantidas para cada recurso distribuído na rede e as atividades de monitoração são realizadas localmente. A base de informação de gerenciamento deve então conter informações para descrever as interfaces ATM e as conexões estabelecidas por cada comutador ATM.

7.1.4 Sub-rede

Uma sub-rede se diferencia de uma camada de rede pelo tipo do ponto de acesso que, no caso de uma sub-rede, é representado pelo ponto de terminação de conexão. Uma sub-rede descreve a estrutura interna de uma camada de rede e pode ser composta por sub-camadas menores e seus respectivos enlaces de interconexão. Tal refinamento deriva em uma sub-rede mínima, que pode então ser mapeada para um recurso de rede. No caso do modelo TMN, as sub-redes são interconectadas através de enlaces que englobam todas as conexões entre duas sub-redes.

7.2 Descrição de aspectos de conectividade

Após identificados os componentes que constituem uma partição de rede, o próximo aspecto a ser descrito envolve as atividades de conexão que associam todas as entidades contidas em uma rede e são responsáveis pela transferência de informação entre conexões pares ou pontos de terminação de *trail*. Tais atividades consistem basicamente na configuração de parâmetros que serão empregados para o estabelecimento de uma conexão de sub-rede ou conexão de enlace.

7.2.1 Aspectos de conexão, enlace e *trail*

Como mencionado anteriormente, a entidade de transporte *trail* se estende ao longo da camada de rede e, em uma abordagem cliente/servidor, é empregada como servidora para as conexões estabelecidas na camada cliente. Neste contexto,

de acordo com o domínio da conexão, dois tipos de conexão são utilizadas para representar uma conexão de sub-rede ou uma conexão de enlace. No primeiro caso, a conexão é mantida nos limites de uma sub-rede e conecta pontos de referência que podem ser configurados e estabelecidos por processos de gerenciamento de conexão. No segundo caso, a conexão é localizada entre duas sub-redes e conecta pontos de referência que podem ser estabelecidos independentemente de um processo de gerenciamento [G.803].

7.2.2 Principais componentes de um fragmento de conectividade

As entidades de transporte contidas no fragmento de conectividade assim como os pontos de referência para a devida conexão estão ilustrados na figura 7.2. O Exemplo apresenta duas sub-redes no qual são caracterizadas conexões de enlace e sub-rede.

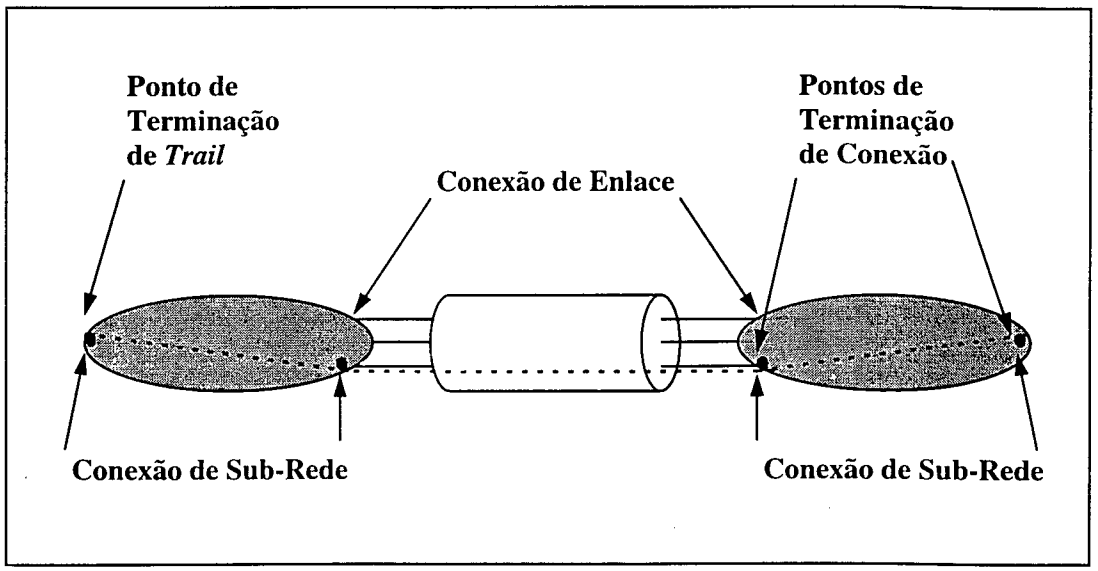

Figura 7.2: Principais componentes do fragmento de conectividade.

Em acréscimo, os pontos de referência responsáveis pela adaptação da saída de uma função de transporte com a entrada de uma outra função de transporte são identificados pelos Pontos de Terminação de *Trail* (*Trail Termination Point*) e pelos

Pontos de Terminação de Conexão (*Connection Termination Points*) localizados nos limites das respectivas camadas de rede e sub-rede.

Um conjunto de características para a modelagem de conexões foram propostas pelo modelo TMN, o sub-conjunto destes aspectos que descrevem uma conexão são herdados de objetos existentes na recomendação M.3100 e são principalmente empregados na configuração de entidades de transporte de conexão. Estas características basicamente envolvem informações que identificam o *trail* subjacente que irá atuar como um servidor para a conexão que será estabelecida na camada cliente.

Tabela II: Conceitos TMN e SNMP para o fragmento de conectividade.

	TMN	SNMP
Conexão de Sub-Rede	A recomendação M.3100 apresenta um objeto *Fabric* que contém objetos de cros-conexão que fazem referência a pontos terminais de conexão.	Grupo de cross-conexão VP/VC para redes ATM.
Conexão de Enlace	Um conjunto de conexões entre duas sub-redes.	Não se aplica.
Trail	Um objeto denominado *Trail* é empregado para a transferência de informações características entre pontos terminais de uma camada de rede.	Não se aplica.

A tabela II descreve os principais componentes relacionados ao estabelecimento de uma conexão. No caso do modelo TMN, este apresenta diferentes tipos de objetos para a manipulação dos diferentes tipos de conexões. Como consequência, a modelagem dos componentes presentes em uma camada de rede não está restrito a abstração dos pontos de referência que identificam os

pontos de terminação de origem e destino de uma conexão ou um *trail* mas é estendido aos aspectos gerenciáveis de uma camada de rede que representam a associação entre pontos de terminação [KoRo97b].

7.2.2.1 Conexão de sub-rede

A visão TMN de uma conexão de sub-rede, denominada de *cross-connection*, envolve um conjunto de pontos de terminação, a associação entre estes pontos e uma classe para o estabelecimento e a liberação de uma conexão. Informações são mantidas para identificar o limite dos pontos de terminação, o tipo de sinal que está sendo manipulado e a direção do fluxo de informação.

O modelo SNMP representa uma conexão de sub-rede no seu nível mais detalhado de abstração no qual objetos são diretamente mapeados em um comutador ATM. A informação é restrita à identificação do par de portas localizadas no comutador e à identificação dos identificadores de Canal Virtual e Caminho Virtual. O modelo SNMP não fornece suporte para uma abstração mais elevada da rede e a representação fim-a-fim de uma conexão de sub-rede.

7.2.2.2 Conexão de enlace

Uma conexão de enlace no modelo TMN é formada pelo conjunto total de conexões entre duas sub-redes. Um objeto de conexão mantém ponteiros de referência para um objeto de *trail* servidor, diferentes conexões entre um par de sub-redes podem referenciar *trails* distintos.

8. Aspectos de Gerenciamento de Redes ATM

O modelo TMN definido pelo ITU-T propõe uma arquitetura de sistema de operação distribuída com interfaces de operação padronizadas entre os componentes de gerenciamento. Tais componentes empregam o protocolo CMIP para a troca de operações e são modelados por um modelo de informação totalmente orientado a objetos o qual é estruturado em três hierarquias de herança, nomeação e registro. Sistemas gerentes são representados por sistemas de operação OS que são responsáveis pela monitoração e controle dos elementos de redes distribuídos em uma rede. Ambos os componentes, sistemas de operação e elementos de rede, são conectados através de uma rede de comunicação de dados encarregada pela adequada troca de informações. Em acréscimo, um conjunto de interfaces é também definido para a comunicação inter e intra-TMN [KoRo97a] [KoRo97b].

8.1 Modelos de Informação

O organismo de padronização ITU-T recentemente propôs um conjunto de recomendações contendo novas funções para o gerenciamento das camadas de Canal Virtual e Caminho Virtual, respectivamente VC (*Virtual Channel*) e VP (*Virtual Path*). O modelo de informação especificado nestas recomendações basicamente descrevem classes de objetos gerenciados que herdam características definidas nas classes de objetos propostas nas recomendações M.3100 *Generic Management Information Model* [M.3100], Q.821 *Stage 2 e Stage 3 Description for the Q3 Interface: Alarm Surveillance* [Q.821] e Q.822 Stage 1, *Stage 2 e Stage 3 Description for the Q3 Interface: Performance Management* [Q.822]. A coleção destes objetos gerenciados fornece informações de gerenciamento que podem ser adotadas por

elementos de rede ATM e define funções de gerenciamento VP e VC para a camada ATM, como também um modelo de informação para a subcamada de convergência de transmissão.

O modelo de informação genérico especificado na recomendação M.3100 contém classes de objetos gerenciados necessários para descrever a troca de informações através das interfaces definidas na arquitetura TMN. Este modelo de informação identifica principalmente classes de objeto gerenciados que são comumente adotados no gerenciamento de redes de telecomunicações e é genérico o suficiente para ser utilizado no gerenciamento de rede independentemente da capacidade tecnológica dos equipamentos de comunicação envolvidos. Tal característica garante a viabilidade de conceitos TMN para a modelagem e o subseqüente gerenciamento de componentes específicos em uma rede ATM.

8.2 Serviços de Configuração

Uma abordagem inicial para a construção de serviços de gerenciamento consiste no fornecimento de um mecanismo de notificação que suporte a emissão de relatórios que objetivam informar o estado corrente do Elemento de Rede que está sendo controlado. Neste contexto, notificações podem informar a aplicação de gerenciamento quando o Elemento de Rede foi inicializado e está então disponível para operações subseqüentes. Estas notificações necessitam apenas fornecer uma indicação de que a inicialização do Elemento de Rede ATM foi concluída. Após as informações de gerenciamento tenham sido coletadas e o estado operacional do Elemento de Rede tenha sido analisado, demais operações podem ser realizadas com o objetivo de atualizar os parâmetros de controle para as atuais e futuras demandas do fluxo da rede [KoRo97a] [KoRo97b].

O relacionamento entre os elementos envolvidos no estabelecimento de conexões de Canais Virtuais e Caminhos Virtuais estão ilustrados na figura 8.1. A Camada de Canal Virtual contém uma função de adaptação para o ponto de acesso a *trail* e a subseqüente função de terminação de *trail* responsável pela adaptação da porta de entrada/saída do pondo de terminação de *trail* VC e o correspondente ponto de terminação VC. Procedimentos similares são realizados pela Camada de Caminho Virtual para a configuração dos parâmetros associados com o estabelecimento de conexões VP.

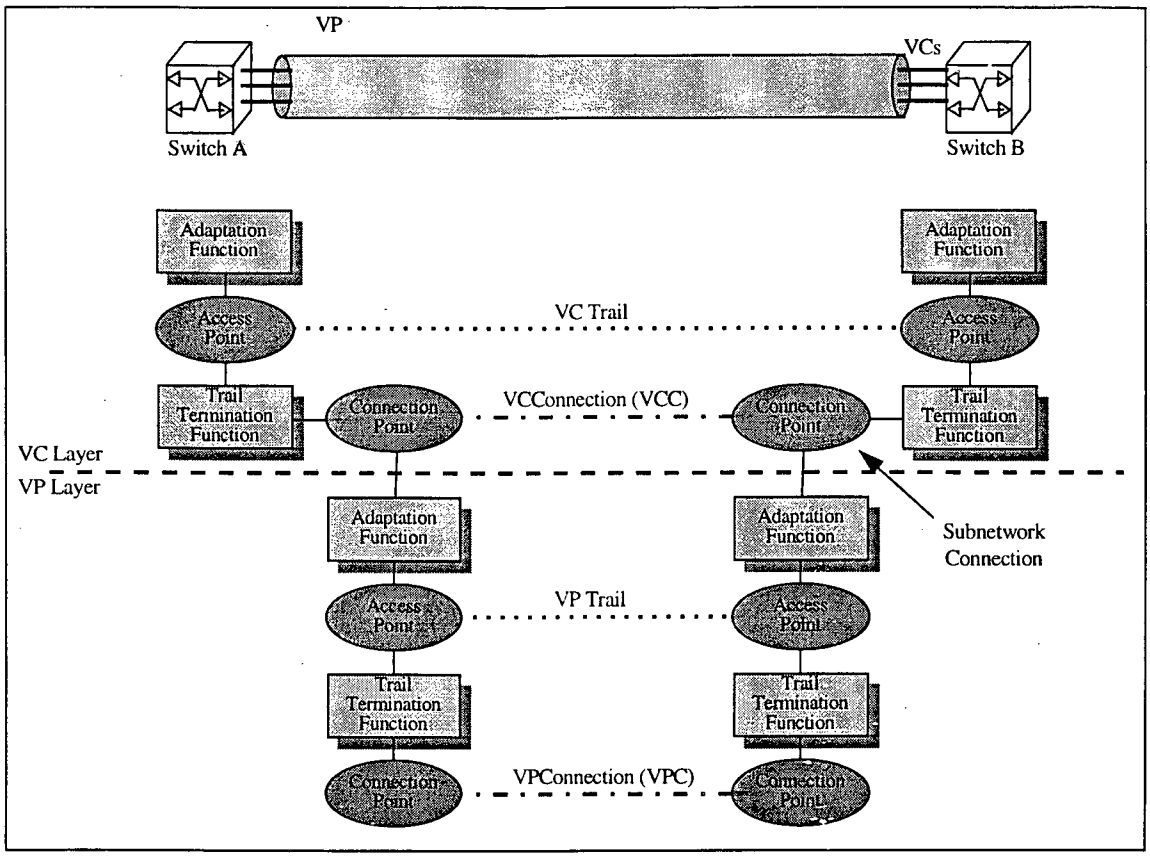

Figura 8.1: Modelo de conexões VP e VC entre dois comutadores ATM.

Parâmetros relevantes relacionados a configuração de interfaces UNI e NNI podem ser alterados para o adequado funcionamento dos componentes ATM. Tais informações definem principalmente o identificador de interface, o ponto de

terminação de trilha subjacente, o número máximo de conexões da Canais Virtuais e Caminhos Virtuais ativos e o número the bits VPI e VCI alocados. O último é essencial para adaptar restrições de endereçamento para os baixos valores suportados de VPI e VCI do equipamento em cada ponto final da interface. No caso de células UNI e NNI, dois tipos de processamento de VP e VC são realizados de acordo com a funcionalidade final da interface da célula. Funções de configuração devem identificar a interface da célula e alocar apropriado espaço de bit VPI, tal funcionalidade atende um requisito das células NNI por um maior número de identificador VPI.

Atividades subseqüentes associadas com o estabelecimento de uma conexão fim-a-fim envolve a configuração dos pontos de terminação das conexões VP e VC nas portas de entrada e saída do Elemento de Rede ATM. Os parâmetros estabelecidos serão utilizados para configurar a conexão de ambos os pontos terminais VP e VC e manter o fluxo de informação. Funções de configuração devem então possibilitar a criação dos correspondentes pontos de terminação de conexão VP e VC através da identificação das portas de entrada e saída das respectivas conexões VP e VC a serem estabelecidas. Parâmetros de enlace VP e VC podem ser alterados a qualquer momento entre o estabelecimento e a liberação de uma conexão garantindo, desta forma, a flexibilidade na configuração da rede.

Procedimentos similares também devem ser adotados em relação ao estabelecimento e a liberação de pontos de terminação de *trail*. Neste caso, funções de configuração são requisitadas para identificar os pontos terminais de uma conexão de enlace VP e VC e instanciar a função de terminação de *trail* VP e VC. Funções adicionais também devem ser fornecidas com o objetivo de unir a porta de entrada e da função de terminação de *trail* VP e VC com o ponto terminal de

conexão de enlace VP e VC. Tais parâmetros devem ser modificados para refletir o estado corrente do fluxo de células.

Procedimentos adicionais devem ser adotados para o estabelecimento e a liberação de conexões VP e VC e fornecer a funcionalidade para reservar, criar e liberar as conexões em questão e desalocar os correspondentes recursos.

8.3 Modelo de Informação TMN

Durante a construção, realizada neste trabalho, dos objetos presentes no modelo de informação M.3100 (vide anexo 11.5), tratamento especial foi adotado para as classes que seriam posteriormente empregadas pelo Elemento de Rede ATM. Tais classes são utilizadas, entre outras facilidades, para modelar aspectos lógicos que possibilitem a inicialização de um Elemento de Rede, fornecer mecanismos para a manipulação de eventos e identificar os componentes básicos de um Elemento de Rede. Neste contexto, classes de objetos podem representar diferentes agrupamentos contidos em um mesmo Elemento de Rede através da abstração de redes, pontos de terminação e detalhes específicos associados com técnicas de comutação e transmissão. A hierarquia de herança do modelo de informação de gerenciamento contém em seus ramos, um conjunto de objetos gerenciados que fornece atributos, operações e notificações para a implementação de um Elemento de Rede para o gerenciamento de pontos de terminação de *trails* e conexões requisitados durante o estabelecimento de uma conexão. Em virtude da característica de que conexões de Canais Virtuais e Caminhos Virtuais são definidos por parâmetros configuráveis, tais parâmetros e atividades correlatas podem ser controladas por um sistema de gerenciamento de acordo com o estado atual da rede.

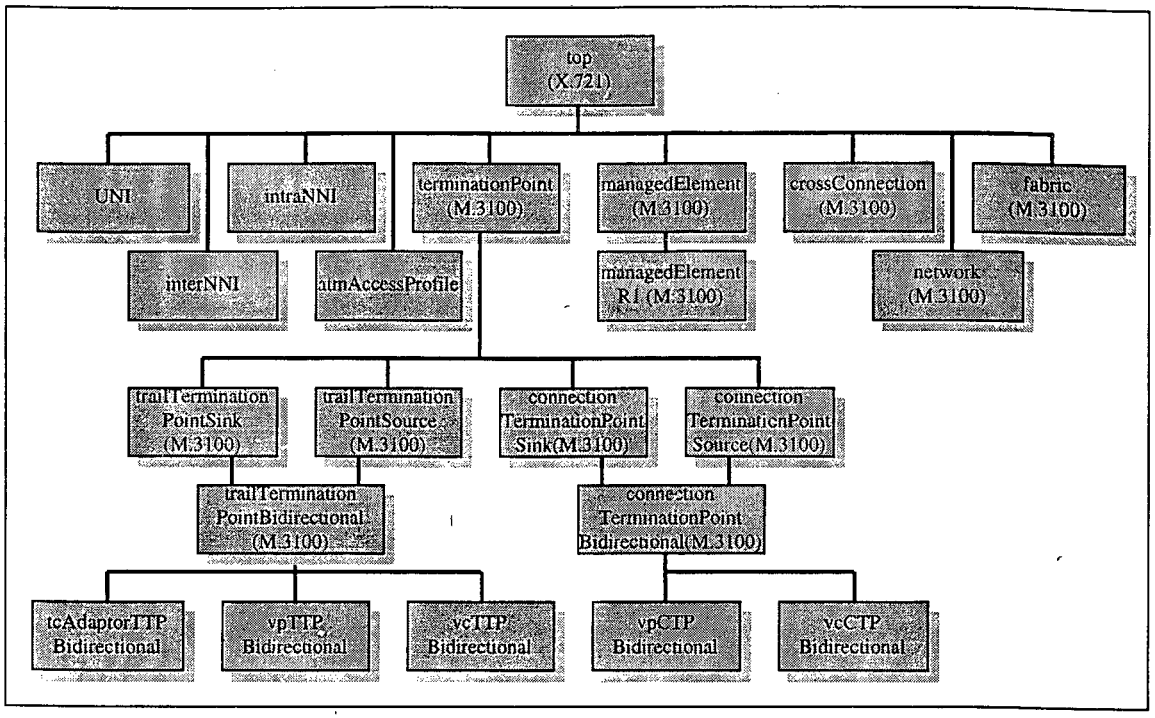

Figura 8.2: Hierarquia de herança do modelo de informação G.atmm.

O modelo de informação apresentado na figura 8.2 ilustra parte da hierarquia de herança dos objetos associados com a configuração de conexões VP e VC. Estes objetos realizam a construção de um Elemento de Rede ATM com o suporte de construções especificadas na recomendação M.3100. Tais classes de objetos modelam características genéricas que são sujeitas a serem empregadas em técnicas de transmissão de rede distintas como no caso da tecnologia ATM (vide anexo 11.10).

TerminationPoint é a superclasse de todos os objetos gerenciados que representam o ponto de terminação de *trail* ou o ponto de terminação de uma conexão de uma entidade de transporte. Os pacotes condicionais e obrigatórios incluídos nesta classe identificam principalmente parâmetros de controle e notificações associadas com a criação e remoção de um objeto gerenciado como

também qualquer alteração no valor de seus atributos. O nível subsequente na hierarquia de herança consiste nas classes de objetos *trailTerminationPointSource* e *trailTerminationPointSink* cujo o principal objetivo é a identificação de um ponto de acesso onde um *trail* origina e termina. Funcionalidades mais complexas são direcionadas aos objetos relacionados com o estabelecimento e liberação de uma conexão. Neste contexto, as classes *connectionTerminationPointSource* e *connectionTerminationPointSink* fornecem os métodos para iniciar e terminar uma conexão de enlace. Em acréscimo, pacotes adicionais contém estruturas que informam o número do canal e o identificador do ponto de terminação de uma conexão.

Métodos para a criação e terminação de um ponto terminal de um *trail* podem ser derivados em uma única classe de objeto *trailTerminationPointBidirectional* através da herança múltipla das correspondentes superclasses de origem e destino. Uma analogia também pode ser realizada para a classe de objeto *connectionTerminationPointBidirectional* cujas principais características são herdadas das classes de objeto *connectionTerminationPointSource* e *connectionTerminationPointSink*.

8.4 O Modelo de Informação do Elemento de Rede ATM

Objetos específicos para a gerência de configuração de conexões VC e VP implementada neste trabalho estão em conformidade com as especificações presentes na recomendação ITU-T G.atmm [G.atmm]. Mais especificamente foi enfatizado o fornecimento de métodos para a conexão bidirecional requerida em transmissões ponto-a-ponto. Neste contexto, a classe *vcTTPBidirectional* é empregada para delimitar a conexão de canal virtual e fornecer operações para solicitar o ponto de terminação, inserir uma célula OAM no fluxo de dados e relatar

se a célula retornou no tempo requerido. A informação fornecida pelo Elemento de Rede deve identificar, entre outras características, o ponto de inserção da célula, o local do retorno da célula e a indicação se a célula está restrita a um segmento ou se atinge a amplitude fim-a-fim da conexão.

Comportamento similar é apresentado pela classe *vpTTPBiriectional* no qual as mesmas ações são realizadas para células de teste OAM dentro do escopo do ponto de terminação de *trail* de um Caminho Virtual. Funções adicionais envolvendo o ponto de terminação de *trail* estão presentes na classe *tcAdaptorTTPBidirectional* cujo objetivo principal consiste na representação do ponto no qual ocorre a adaptação da camada ATM com a estrutura do meio físico subjacente.

Em conformidade com as definições propostas na recomendação [G.atmm] a classe de objeto *vcCTPBidirectional* apresenta os métodos apropriados para delimitar enlaces de Canal Virtual. Os atributos inseridos nesta classe basicamente descrevem o valor VCI, a coleção de descritores de tráfego e a classe de Qualidade de Serviço atribuída à terminação de enlace de Canal Virtual sendo representado. Uma característica relevante desta classe consiste na atribuição do valor VCI da conexão de enlace ao identificador do ponto de terminação de enlace de Canal Virtual, tal valor é empregado pelo objeto para a construção da hierarquia de *containment*. Em acréscimo, atributos para a representação de medidas de performance como, por exemplo, taxa pico de fluxo de células, taxa sustentável de célula, tolerância a rajadas, tolerância de atraso de células e classe de Qualidade de Serviço também fazem parte das atividades de monitoração mantidas pelo objeto. Sob a perspectiva do gerencimento de falhas, esta classe de objetos também apresenta um pacote condicional para o devido teste de conexão de enlace. O conjunto de atributos, operações e comportamento mencionados na classe

vcCTPBidirectional foram mantidos na classe *vpCTPBidirectional* para realizar funções no âmbito da sub-camada VP. Tal característica garante o uso de objetos separados com processamento similar para as diferentes sub-camadas de Canal Virtual e Caminho Virtual contidas na camada ATM.

Após configurados os pontos de terminação de conexão e *trail* em ambas as sub-camadas VC e VP, a classe de objeto *atmFabric* apresentada na figura 8.2, pode ser criada para efetivamente controlar o estabelecimento e a liberação de uma conexão ATM. Neste caso, os pontos de terminação a serem conectados ou disconectados são identificados pela especificação dos respectivos objetos *vcCTPBidirectional* e *vpCTPBidirectional* ou pela especificação das características dos pontos finais de terminação. Informações adicionais devem ser fornecidas com o objetivo de configurar o Elemento de Rede para atuar como uma interface intra-UNI, inter-UNI ou UNI, neste caso respectivas classes de objeto denominadas *intra-NNI*, *inter-NNI* e *UNI* são responsáveis em identificar o sistema gerenciado pelo fornecimento de uma referência ao objeto *tcAdaptorTTPBiDirectional* para a devida representação do ponto onde ocorre a adaptação da camada ATM com a infraestrutura subjacente.

Finalmente, como apresentado na figura 8.2, a classe de objeto *atmAccessProfile* é empregada para identificar o número máximo de VPCs e VCCs qua podem estar simultaneamentes ativos enquanto uma conexão é mantida, como também o número máximo de bits qua podem ser alocados nos campos de identificadores de VPI e VCI.

8.5 Aspectos de Construção do Elemento de Rede ATM

Como mencionado anteriormente, na plataforma OSIMIS os atributos são representados por meio de classes de objeto em linguagem C++. Cada classe de objeto faz referência a métodos genéricos fornecidos pelos mecanismos de suporte à manipulação da sintaxe ASN.1. Estes métodos implementam os aspectos para consultar e alterar os valores de atributos, como também codificar (uma estrutura em C para um elemento de apresentação ASN.1) e decodificar (um elemento de apresentação ASN.1 para uma estrutura em C) os valores atribuídos [RoCo96].

Como mencionado no Quarto Capítulo, os atributos são representados por meio de classes de objetos, conseqüentemente a especificação ASN.1 associada com a sintaxe de atributos deve ser implementada de acordo com o paradigma de orientação a objetos. A primeira etapa para a implementação dos atributos consiste na geração automática de especificações ASN.1 (vide anexos 11.6 e 11.11) para estruturas C pelo compilador ASN.1 fornecido pela ambiente de desenvolvimento ISODE. Um sub-conjunto das estruturas C geradas podem ser observadas na segunda coluna apresentada nas tabelas III e IV. Após a construção das estruturas de dados são implementadas as funções para a manipulação da sintaxe ASN.1 (vide anexos 11.9 e 11.13).

A etapa final para a construção de um atributo consiste no desenvolvimento das classes de objeto C++ para a representação apropriada dos aspectos semânticos dos atributos (vide anexos 11.8 e 11.12). A terceira coluna das tabelas III e IV identificam as novas classes de atributos implementadas neste trabalho.

Tabela III: Classes de atributos da recomendação G.atmn.

Sintaxe do atributo	Estrutura em linguagem C	Classe C++
BurstTolerance	type_AtmMIBMod_BurstTolerance*	BurstTolerance
CDVTolerance	type_AtmMIBMod_CDVTolerance*	CDVTolerance
PeakCellRate	type_AtmMIBMod_PeakCellRate*	PeakCellRate
QosClass	type_AtmMIBMod_QosClass*	QosClass
SustainableCellRate	type_AtmMIBMod_SustainableCellRate*	SustainableCellRate
OAMPeakCellRate	type_ASN1TypeModule_OAMPeakCellRate*	OAMPeakCellRate

Tabela IV: Classes de atributos do padrão M.3100.

Sintaxe do atributo	Estrutura em linguagem C	Classe C++
AlarmStatus	type_ASN1DefinedTypesModule_ AlarmStatus*	AlarmStatus
CurrentProblemList	type_ASN1DefinedTypesModule_ CurrentProblemList*	CurrentProblemList
SystemTimingSource	type_ASN1DefinedTypesModule_ SystemTimingSource*	SystemTimingSource
SystemTitle	type_ASN1DefinedTypesModule_ SystemTitle*	SystemTitle
ObjectList	type_ASN1DefinedTypesModule_ ObjectList*	ObjectList
ConnectivityPointer	type_ASN1DefinedTypesModule_ ConnectivityPointer*	ConnectivityPointer
CrossConnectionObjectPointer	type_ASN1DefinedTypesModule_ CrossConnectionObjectPointer*	CrossConnectionObjectPointer
DownstreamConnectivityPointer	type_ASN1DefinedTypesModule_ DownstreamConnectivityPointer*	DownstreamConnectivityPointer
SupportableClientList	type_ SMI_ObjectClassList*	SupportableClientList
PointerOrNull	type_ASN1DefinedTypesModule_ PointerOrNull*	PointerOrNull

Os tipos de atributos *Alarm Status* e *Current Problem List* são utilizados para representar aspectos de gerenciamento de falhas e descrevem respectivamente o grau de severidade do alarme emitido por um objeto e identifica falhas correntes que

podem ser observadas por um Elemento de Rede. A informação contida no atributo *Current Problem List* inclui uma ampla variedade de alarmes que podem variar de uma simples notificação sobre a inexistência de uma instância de objeto até tipos de notificações mais complexas como a identificação de dois pontos terminais recentemente conectados.

Tabela V: Sintaxes predefinidas existentes na plataforma OSIMIS.

Atributo	Sintaxe	Recomendação
oamEgressCDVTolerance	Integer	G.ATMM
oamIngressCDVTolerance	Integer	G.ATMM
segmentEndPoint	Boolean	G.ATMM
tcTTPId	SimpleNameType	G.ATMM
vcCTPId	SimpleNameType	G.ATMM
vcTTPId	SimpleNameType	G.ATMM
vpCTPId	SimpleNameType	G.ATMM
vpTTPId	SimpleNameType	G.ATMM
channelNumber	Integer	M.3100
characteristicInformation	OID	M.3100
cTPId	SimpleNameType	M.3100
locationName	GraphicString	M.3100
managedElementId	SimpleNameType	M.3100
networkId	SimpleNameType	M.3100
networkLevelPointer	ObjectInstance	M.3100
tTPId	SimpleNameType	M.3100
userLabel	GraphicString	M.3100
vendorName	GraphicString	M.3100
version	GraphicString	M.3100

Os seguintes tipos de atributos *Connectivity Pointer*, *Cross Connection Object Pointer* e *Down Stream Connectivity Pointer* requisitados no Elemento de Rede identificam principalmente as Instâncias de objeto que representam o ponto de terminação de uma conexão e um *trail.*

Parâmetros de performance existentes na rede são requisitados pela recomendação G.atmm e descrevem informações relacionadas a taxa máxima de células bem como o fluxo sustentável de células a serem transmitidas. A Qualidade de Serviço também é assegurada através da identificação da classe de serviço sendo fornecida.

A tabela V apresenta um conjunto de atributos adicionais requisitados pelo Elemento de Rede ATM o qual faz uso de algumas classes pré-definidas presentes na plataforma OSIMIS. O sub-conjunto destes atributos caracterizado pela sintaxe *Simple Name Type* é basicamente empregado para identificar as instâncias de objeto contidas na hierarquia de *containment*. Alguns atributos distintos são utilizados em aspectos diferentes do Elemento de Rede e podem indicar se a instância do objeto foi configurada para representar um segmento da conexão ou indicar um eventual atraso de células. Um atributo adicional é também fornecido para especificar o número do canal da especificação.

Finalmente, atributos opcionais foram implementados para informar sobre o local onde o Elemento de Rede foi instalado, o nome do fabricante do recurso sob controle, a versão do equipamento e um nome para identificar o recurso.

Ações são realizadas pelo objeto *Fabric* com o objetivo de controlar o estabelecimento e a liberação de uma conexão ponto-a-ponto tanto para a camada de Caminho Virtual como para a camada de Canal Virtual. A sintaxe implementada na ação contém um conjunto de informações que fazem referência a instâncias de objetos específicas que podem refletir uma conexão UNI, intra-UNI ou inter-UNI, em acréscimo, argumentos adicionais são empregados para descrever parâmetros de performance. Importante enfatizar, que o comportamento do Elemento de Rede em

construção está basicamente incorporado nos atributos e ações anteriormente mencionados.

8.6 Ambiente de Rede ATM sob Análise

Esta seção descreve o ambiente de teste proposto pela equipe da Divisão de Gerência de Redes do Núcleo de Processamento de Dados da Universidade Federal de Santa Catarina (UFSC) como uma abordagem inicial para a construção de um backbone ATM com o objetivo de interconectar o conjunto de sub-redes contidas na rede UFSC.

A figura 8.3 ilustra a topologia da rede na qual estações de trabalho são interconectadas por comutadores ATM. O equipamento ATM disponível na UFSC é composto de dois comutadores ATM IBM 8260 [IBM 8260] e de um comutador ATM *Bay Network Centillion* 100, também fazem parte deste conjunto de equipamentos placas ATM instaladas em estações de trabalho e PCs.

Os comutadores estão interconectados através de interfaces NNI (*Network-to-Node Interface*) sobre enlaces de 155 e 100 Mbps. Os comutadores IBM 8260 estão interconectados em uma razão de 100 Mbps, enquanto a conexão com o Centillium 100 é realizada a 155Mbs. As interfaces UNI (*User-to-Network Interfaces*) são empregadas apenas para a conexão das placas ATM com os comutadores.

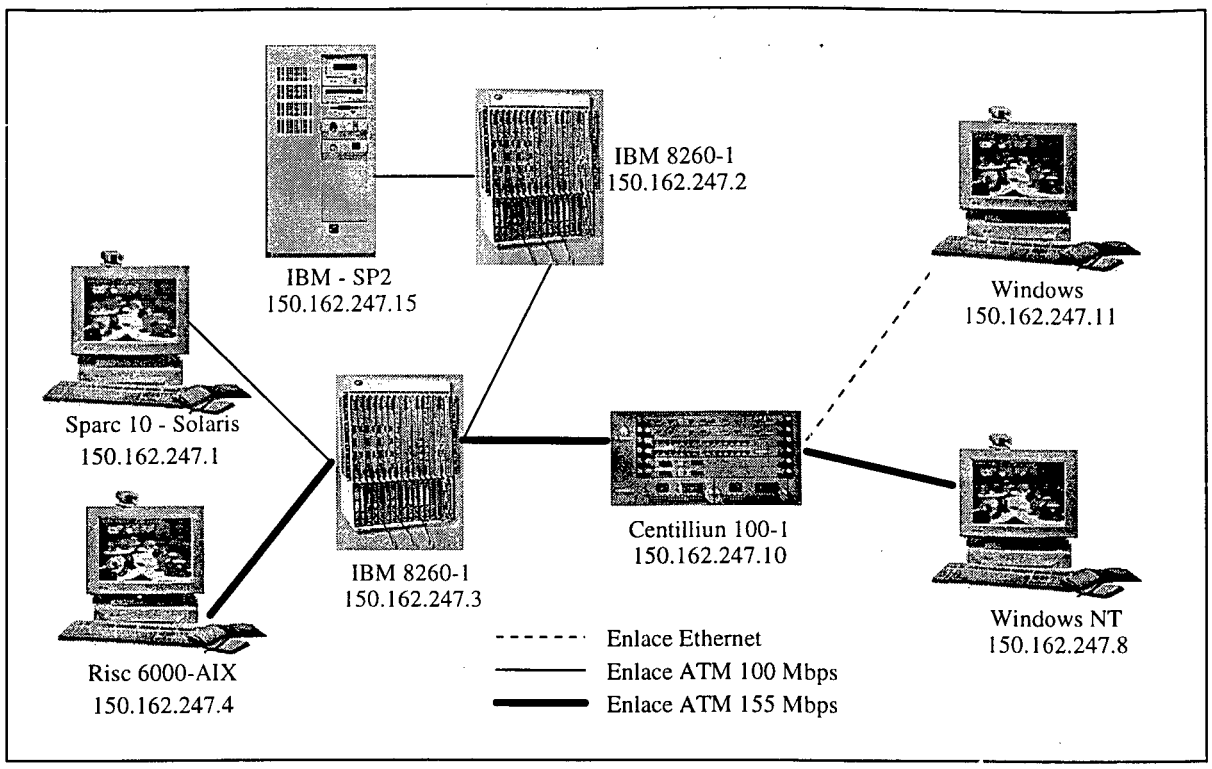

Figura 8.3: Ambiente de Teste ATM Disponível na Rede UFSC.

O comutador IBM 8260 adota ambos os tipos de conexão virtual permanente e comutado, respectivamente PVC (*Permanent Virtual Connection*) e SVC (*Switched Virtual Connection*). O suporte a conexões virtuais permanentes permite o uso the sistemas de gerenciamento externos para a configuração dos parâmetros necessários para o estabelecimento e a liberação de canais e caminhos virtuais. Um sub-conjunto destes parâmetros são utilizados para identificar um ponto terminal de conexão e descrever características relevantes relacionadas a garantias de Qualidade de Serviço e alocação de largura de banda. Tais parâmetros são passíveis de serem configurados por meio de uma MIB II SNMP instalada no comutador e portanto viabilizado o uso de uma plataforma de gerenciamento para o controle de conexões virtuais. Como consequência, tabelas específicas contém valores que identificam a conexão e definem o tipo de interface ATM.

O modelo de informação SNMP incorporado aos comutadores ATM IBM 8260 e Centillium 100 não está em conformidade com a arquitetura TMN, tal característica

implica no uso de um adaptador Q para tornar possível o controle dos comutadores por parte de uma plataforma de gerenciamento CMIP. Embora a platforma OSIMIS forneça um *gateway* CMIP/SNMP que implementa a funcionalidade inerente a um adaptador Q, métodos adicionais devem ser acrescentados ao adaptador-Q com o objetivo de implementar o mapeamento da informação necessária ao Elemento de Rede ATM.

9. Conclusões e Trabalhos Futuros

Apesar dos esforços dispendidos com o objetivo de atender as padronizações existentes em modelos de gerência existentes nas arquiteturas TMN, OSI e Internet, a análise das principais plataformas de gerenciamento existentes no mercado revelou uma elevada diversidade de propostas com o objetivo de possibilitar a monitoração e o controle da rede.

O desenvolvimento de aplicações não está restrito unicamente ao protocolo de informação, diversos outros fatores devem ser considerados durante a construção de novos serviços de gerência. Neste contexto, as interfaces de acesso ao protocolo de informação podem simplificar os procedimentos utilizados para a implementação de processos de gerenciamento. Em acréscimo, o emprego de linguagens de especificação, e por seguinte, a utilização de compiladores para a devida conversão em código de implementação são capazes de reduzir consideravelmente as atividades dos projetistas de aplicação. Outro fator determinante no desenvolvimento de serviços consiste nos mecanismos de interface com o usuário os quais, em muitos casos, podem fornecer recursos gráficos para a inclusão de novos componentes de gerenciamento.

A escolha da plataforma OSIMIS como ambiente de desenvolvimento para os serviços de gerência propostos neste trabalho é justificada por esta apresentar um conjunto de interfaces que possibilitam a construção de componentes TMN. Em acréscimo a estas interfaces a plataforma OSIMIS oferece também compiladores para a manipulação das sintaxes ASN.1 e GDMO, implementação dos principais protocolos de gerenciamento CMIP e SNMP e, por seguinte, um *gateway* a nível da camada de aplicação para a interconexão entre ambos os protocolos. Tais

componentes possibilitam ao OSIMIS o controle e a monitoração de MIBs CMIP e SNMP, e expandem a sua área de atuação aos mais diversos ambientes de redes de computadores.

Com o objetivo de realizar pesquisas em ambientes de redes de telecomunicações e explorar as interfaces para a construção de processos de gerenciamento existentes na plataforma OSIMIS foram desenvolvidas extensões a esta plataforma. Como mencionado no Quarto Capítulo, a implementação destes novos serviços envolveu o emprego de parte dos componentes fornecidos pelo OSIMIS e forneceu subsídios para o posterior desenvolvimento de aplicações de gerência.

Em virtude da plataforma OSIMIS ter sido projetada com o objetivo de atuar principalmente em redes de telecomunicações, análises foram feitas com o objetivo de identificar os serviços mais comumente utilizados nestas redes. Em paralelo, uma vez observado que o OSIMIS não oferecia recursos para o desenvolvimento de projetos associados à gerência de contabilização, foi decidido que seriam concentrados esforços com o objetivo de enriquecer a plataforma OSIMIS viabilizando uma estrutura de suporte ao desenvolvimento de aplicações de contabilização.

Como a plataforma de gerenciamento está em conformidade com o modelo de gerenciamento OSI, a melhor abordagem a ser seguida consistiu na implementação dos serviços propostos pela Função de Medida de Contabilização inserida no modelo funcional OSI. Inicialmente foi realizada a implementação dos requisitos associados a esta função, no entanto, durante o desenvolvimento do trabalho, tal recomendação foi atualizada pelos organismos internacionais de padronização. Anteriormente denominada Função de Medida Contabilização [ISO

10164-10a], esta função recebe o título atual de Função de Medida de Uso [ISO 10164-10b]. Esta atualização consiste em uma versão ampla e atende a um escopo mais diversificado de serviços.

A Função de Medida de Uso propõe soluções para as restrições existentes na antiga versão. As informações coletadas no processo de medida de uso, antes restritas a unidade, quantidade consumida e classe de tarifa, atualmente podem ser definidas pelo administrador para a inclusão de quaisquer informações consideradas relevantes. Em acréscimo, a nova função prevê aspectos relacionados a auditoria do processo de contabilização e apresenta processos posteriores ao controle e à coleta de dados.

A extensão à antiga Função de Medida de Contabilização para atender os novos requisitos propostos na Função de Medida de Uso acrescentou à plataforma OSIMIS novas características do serviços de contabilização TMN. Um ponto favorável à utilização desta função é a abrangência de seus atributos e objetos gerenciados. Desta forma é possível, fazendo uso de especializações, contabilizar um grande número de recursos e serviços.

Viabilizada a implementação de uma estrutura de contabilização segundo os padrões definidos pela ISO, pôde-se fazer uso de todos os conceitos e facilidades previstos em sua Estrutura de Informação de Gerenciamento e nos serviços oferecidos pelo CMIS.

Facilmente estendida, como foi observado durante a construção de contabilização, a inclusão de um novo recurso a ser contabilizado é realizado através de objetos gerenciados que o representem logicamente, enviando informações à função de contabilização. Desta forma, por meio da Função de

Medida de Uso implementada, pode-se por incrementos sucessivos, criar uma Base de Informação de Gerenciamento que represente as características específicas de cada recurso.

As atividades realizadas em gerência de contabilização forneceram subsídios para o desenvolvimento de funções complexas que envolvam os recursos do modelo TMN e o emprego de interfaces e ferramentas para a construção de processos de gerenciamento. Foram realizados estudos para a implementação de serviços de gerenciamento em redes de alta velocidade, mais especificamente funções para a configuração de Canais Virtuais e Caminhos Virtuais em ambientes de redes ATM. A abordagem inicial consistiu nc desenvolvimento de um Elemento de Rede TMN que fornecesse os serviços necessários para a configuração e o estabelecimento de conexões de Canais Virtuais e Caminhos Virtuais entre pontos terminais de comunicação.

O ambiente de teste ATM utilizado neste trabalho não apresentava um modelo de informação de gerenciamento que estivesse em conformidade com o modelo TMN, desta forma as atividades de desenvolvimento de serviços de gerência não estiveram restritas à construção do Elemento de Rede TMN. Uma parte do desenvolvimento da aplicação esteve direcionada ao mapeamento das informações requisitadas pelo Elemento de Rede e a MIB SNMP existente no comutador ATM. Esta interface de um recurso não TMN com o Elemento de Rede utilizou um adaptador Q existente na plataforma OSIMIS. Este adaptador foi então alterado para comportar a interface do Elemento de Rede com o comutador ATM. Desta forma, a contribuição desta etapa do trabalho não esteve restrita a extenção da plataforma OSIMIS pela implementação de um Elemento de Rede, mas também forneceu benefícios em aspectos de relacionados ao mapeamento de informações entre dois

modelos de gerenciamento distintos, mais especificamente os modelos SNMP e TMN.

Os serviços de gerenciamento em desenvolvimento neste trabalho fornecem a funcionalidade para o gerenciamento de conexões de Canal Virtual e Caminho Virtual e servem de suporte para a construção de novas funções de gerenciamento de redes ATM envolvendo aspectos associados com alocação de largura de banda, monitoração de tráfego, e uma variedade de serviços adicionais relacionados com a gerência de segurança e contabilização. Informações sobre a configuração da conexão de Canais Virtuais e Caminhos Virtuais podem então ser obtidos a partir da base de informação de gerenciamento. Tal procedimento reduz os custos para a construção de aplicações de gerência em virtude de que parte do comportamento da aplicação de gerenciamento pode ser realizada pelo Elemento de Rede.

10. Bibliografia

[AyTa 96] Aydemir, B.; Tanzini, J. ***An ATM Network-View Model for Integrated Layer-Network Management.*** IEEE/IFIP 1996 Network Operations and Management Symposium, Kyoto - Japão, abril 1996.

[BRISA] BRISA - **Gerenciamento de Redes: Uma Abordagem de Sistemas Abertos**. Editora Makron Books. 1993.

[BuGo95] Bussmann, L.A.S.; Gonçalves, M.M.; Mayrhofer, R.V.; Ruiz, L.B. **Análise sobre a Plataforma HP - OpenView**. Relatório de Pesquisa - Projeto PLAGERE, CEFET-PR, abril 1995.

[CORBA rv2.0] ***The Common Object Request Broker: Architecture and Specification***, Revision 2.0, julho 1995.

[CoRo95] Corrêa Neto, A.S.; Rogerio, K.O. **Implementação da Função de Medida de Uso e Contabilização de Recursos para a Plataforma de Gerenciamento OSIMIS/ISODE**. Relatório de Pesquisa - Projeto PLAGERE, UFSC, outubro 1995.

[CoRo96] Corrêa Neto, A.S.; Rogerio, K.O.; Kormann, L.F.; Westphall, C.B. **Implementação de novos serviços de contabilização para a plataforma OSIMIS.** XXIII Seminário Integrado de Software e Hardware, Recife, agosto 1996.

[FaMa95] Farias, L.J.L.; Martins, J.S.B. **Análise da plataforma de gerenciamento SunNet Manager**, Relatório de Pesquisa - Projeto PLAGERE, UFPB, abril 1995.

[Fark93] Farkouh, S.C. ***Managing ATM-Based Broadband Networks***, IEEE Communications Magazine, maio 1993.

[FeLo95] Ferreira, A.R.Q.; Lorena, P.S. **Desenvolvimento de aplicações e novas funções de gerência na plataforma ISM/Bull.** Relatório de Pesquisa - Projeto PLAGERE, CPqD - Telebrás, 1995.

[FuYo 96] Fujii, N.; Yoda, I, ; Minato, K. ***An ATM Transport Network Operation System Based on TMN.*** IEEE/IFIP 1996 Network Operations and Management Symposium, Kyoto - Japão, abril 1996.

[G803] ***ITU-T Recommendation G.803 - Architectures of Transport Networks on the Synchronous Digital Hierarchy***, março 1993.

[GDMO93] ***OSIMIS GDMO Compiler User Manual.*** Department of Computer Science, University College of London, 1993.

[Ghet95] Ghetie, J. ***Management Platforms Analysis and Evaluation - Tutorial*** Fourth IFIP/IEEE International Symposium on Integrated Network Management, maio E.U.A, 1995.

[IBM 8260] ***IBM 8260 Multiprotocol Intelligent Switching Hub - ATM Control Point and Switch Module Installation and User's Guide***, março 1995.

[IS 9595] **ISO/IS 9595:** ***Information Technology - Open Systems Interconnection - Common Management Information Service Definition***, julho 1991.

[IS 9596] **ISO/IS 9596:** ***Information Technology - Open Systems Interconnection - Common Management Information Protocol Specification, Version 2***, julho 1991.

[IS0 8824] **ISO 8824:** ***Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntex Notation One (ASN.1)***, novembro 1987.

[ISO 10164-6] **ISO 10164-6:** ***Information Technology - Open Systems Interconnection - Systems Management Functions - Part 6: Log Control Function***, marco 1991.

[ISO 10164-10a] ***Information Technology - Open Systems Interconnection - Systems Management Functions - Part 10: Accounting Meter Function***, março 1993.

[ISO 10164-10b] ***Information Technology - Open Systems Interconnection - Systems Management Functions - Part 10: Usage Metering Function***, outubro 1994.

[ISO 10165-4] ***Information Technology: Open Systems Interconnection Structure of Management Information - Part 4: Guidelines for Definition of Managed Objects***, março 1991.

[KoCo94a] Kormann, L.F.; Coser, A.; Vieira, E.M.; Westphall, C.B. **Implementação da função de contabilização para a plataforma de gerência OSIMIS.** XXVI Congresso Nacional de Informática e Telecomunicações. Anais em disquete (resumo). Salvador BA, novembro 1994.

[KoCo94b] Kormann, L.F.; Coser, A. **Implementação da Função de Contabilização para a Plataforma de Gerenciamento OSIMIS/ISODE.** Trabalho de Conclusão de Curso em Ciência da Computação, Departamento de Informática e de Estatística, UFSC, dezembro 1994.

[KoCo95a] Kormann, L.F.; Coser, A.; Vieira, E.M.; Westphall, C.B. **Extensão da Plataforma OSIMIS pela Implementação da Função de Medida de Uso.** Anais do XIII SBRC: Simpósio Brasileiro de Redes de Computadores, pg. 371-388. Belo Horizonte MG, maio 1995.

[KoCo95b] Kormann, L.F.; Corrêa, A.S.; Rogerio, K.O.; Westphall, C.B. **Análise sobre a Plataforma OSIMIS.** Relatório de Pesquisa - Projeto PLAGERE, UFSC, agosto 1995.

[KoWe95a] Kormann, L.F.; Westphall, C.B. ***An extension to the OSIMIS Management Platform Through Implementing Accounting Services.*** Sixth IEEE/IFIP Distributed Systems: Operations and Management - DSOM'95. Ottawa -Canadá, outubro 1995.

[KoWe95b] Kormann, L.F.; Westphall, C.B. **Análise comparativa entre plataformas.** Relatório de Pesquisa - Projeto Plagere, UFSC, junho 1995.

[KoWe96] Kormann, L.F.; Westphall, C.B.; Coser, A. ***OSI Usage Metering Function for the OSIMIS management plataform.*** Journal of Network and Systems Management. Plenum Publishing Corporation. Meddletown, In the 3Q96 (Vol. 4, No. 3) Issue, 1996.

[KoRo97a] Kormann, L.F.; Rogerio. K.O. ; Westphall, C.B. ***Construction of a TMN Network Element for the Configuration Management of ATM Networks.*** Anais do XV SBRC: Simpósio Brasileiro de Redes de Computadores, São Carlos, São Paulo, maio 1997.

[KoRo97b] Kormann, L.F.; Rogerio. K.O. ; Westphall, C.B. ***A TMN Network Element for the Configuration Management of ATM Networks: Development Aspects***. Anais do XXIV SEMISH: Seminário Integrado de Software e Hardware, Brasília DF, agosto 1997.

[Ku 96] Ku, B.S. ***Use of TMN for SONET/SDH Network Management.*** IEEE/IFIP 1996 Network Operations and Management Symposium, Kyoto - Japão, abril 1996.

[Lore95] Lorena, P.S. **Concepção e Desenvolvimentos para Gerência em SDH.** Relatório de Pesquisa - Projeto Plagere, CPqD - Telebrás, 1995.

[M3100] ***ITU-T Recommendation M3100 - Generic Management Information Model***, agosto 1995.

[MuMa95] Mussi, C.C; Mazorca, V.N.F. **Desenvolvimento de Novos Processos para Gerência de Redes OSI.** Relatório de Pesquisa - Projeto PLAGERE, UFSC, dezembro 1995.

[PaCo92] Pavlou, G.; Cowan, J.; Crowcroft, J. ***A Generic Management Information Base Browser***, University College London, 1992.

[PaBh93a] Pavlou, G.; Bhatti, S.N.; Knight G. ***Automating the OSI to Internet Management Conversion Through the use of an Object-Oriented Plataform,*** IFIP Conference on LAN and MAN Management, Paris, novembro 1993.

[PaBh93b] Pavlou, G.; Bhatti, S.N.; Knight G. ***The OSI Management Information Service User's Manual***, Version 1.0, University College London, fevereiro 1993.

[PaTi94] Pavlou, G.; Tin, T.; Carr, A. ***High-Level APIs in the OSIMIS TMN Plataform: Harnessing and Hiding,*** University College London, setembro 1994.

[PaKn95] Pavlou, G.; Knight, G.; McCarthy, K.; Bhatti, S. ***The OSIMIS Plataform: Making OSI Management Simple***, Fourth IFIP/IEEE International Symposium on Integrated Network Management. Santa Barbara. E.U.A, 1995. maio 1995.

[PhBr93] Phifer, L.; Brusil; P. ***Incorporating OSI Management Technology into the Marketplace.*** Integrated Network Management, III(C-12), IFIP, 1993.

[QuKu95] Quilante, K.; Kurten, R.; Westphall, C. B. **Novas Funções de Gerência de Segurança usando o AIX NetView/6000.** Relatório de Pesquisa - Projeto PLAGERE, UFSC agosto 1995.

[Q.821] ***ITU-T Recommendation Q.821 - Stage 2 e Stage 3 Description for the Q3 Interface: Alarm Surveillance***, março 1993.

[Q.822] ***ITU-T Recommendation Q.822 - Stage 1, Stage 2 and Stage 3 Description for the Q3 Interface: Performance Management***, março 1993.

[SoMa93] Souza, J.N.; MacCarthy K.; Agoulmine, N.; Pavlou, G.; ***CMIP/SNMPv1 Translation Through Application Level Gateways Using the OSIMIS/ISODE Platform***, novembro 1993.

[RFC1212] ***Request for Comments 1212: Concise MIB Definitions***, março 1991.

[RoCo96] Rogério, K.O.; Corrêa Neto, A.S.; Kormann, L.F.; Westphall, C.B. **Implementação de Novos Serviços de Contabilização para a Plataforma OSIMIS.** Anais do XXIII SEMISH: Seminário Integrado de Software e Hardware. pg. 493-504. Recife, Pernambuco, agosto 1996.

[RoOn91] Rose, M.T., Onions J.P., Robbins C.J., "***The ISO Development Environment Users's Manual Version 7.0***", PSI Inc./X-Tel Services Ltd., julho 1991.

[WeKo96] Westphall, C.B.; Kormann, L.F. ***Usage of TMN Concepts for the Configuration Management of ATM Networks***, em Broadband Strategies and Technologies for Wide Area and Local Access Network, Berlin - Alemanha, 10 - 11 outubro de 1996, Roberto Vercelli, Editor, ISBN 0-8194-2357-2, Proc. SPIE vol. 2953, pp 250-7 (1996).

[Yemi93] Yemini, Y. ***The OSI Network Management Model***, IEEE Communication Magazine, maio 1993.

11. Anexos

11.1 Especificação ASN.1 das Sintaxes de Atributos da Recomendação ISO/IEC 10164-10

```
-- ##############################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: UMFObjId.py - 10164-10
                Descrição: Contém a especificação em ASN1 das sintaxes de atributos associados a
                Recomendação ISO/IEC 10164-10
-- ##############################################################################

UMFObjId DEFINITIONS ::=

PREFIXES build parse print

BEGIN

IMPORTS
    GraphicString, SimpleNameType, ObjectInstanceList, OIDList FROM     SMI {
                joint-iso-ccitt
                ms(9)
                smi(3)
                part2(2)
                asn1Module(2)
                1
        }
        ObjectInstance FROM CMIP {
                joint-iso-ccitt
                ms(9)
                cmip(I)
                version1(1)
                protocol(3)
        }
        TimePeriod FROM         Monitor {
                joint-iso-ccitt
                ms(9)
                function(2)
                part11(11)
                asn1Module2(2)
                0
        }
        UsageInfo FROM UMDataInfo {
                joint-iso-ccitt
                ms(9)
                function(2)
                part10(10)
                asn1Modules(2)
                2
        };


SECTIONS build parse print

 -- AccountbleObjectReference ::= ObjectInstance
 -- AccountbleObjectReferenceList ::= SET of ObjectInstance
        ActionArgument ::= CHOICE {
                selectedObjects   ObjectInstanceList,
```

```
                    -- set of data objects, controlled by the control
                    -- object for which the request is appropriete
        allObjects          NULL
                    -- selects all data objects controlled by control
          -- object
        }


ActionResponse ::= SEQUENCE {
                    -- at least one component shall be present
        success             [0] ObjectInstanceList    OPTIONAL,
        failed              [1] ObjectInstanceList    OPTIONAL,
        indeterminate       [2] ObjectInstanceList    OPTIONAL }


AuditInfo ::= SEQUENCE {
        service             OBJECT IDENTIFIER,
        auditDetails        ANY DEFINED BY service }

ControlInfo ::= SEQUENCE {
        actionResponse                  ActionResponse,
        reportingTriggers               [0] ReportingTriggers          OPTIONAL,
        accountableObjectsReferenceList [1] ObjectInstanceList         OPTIONAL,
        dataObjectsReferenceList        [2] ObjectInstanceList         OPTIONAL,
        additionalInformation           [3] ManagementExtensionList    }

ManagementExtensionList ::= SET OF ManagementExtension

ControlStatusValue ::=    ControlStatus ( WITH COMPONENT (suspended))

DataErrors ::= CHOICE {
        possibleErrors      OIDList,  -- SET OF PossibleError
        noProblem           NULL }

-- DataObjectsReferenceList ::= ObjectInstanceList


DeniedMeteringAction ::= ENUMERATED {
        canNotStart(0),
        canNotSuspend(1),
        canNotResume(2)}

Induced ::= ENUMERATED {
        start(0),
        suspend(1),
        resume(2),
        delete(3),
        disabled(4),
        enabled(5) }

NotificationCause ::= CHOICE {
        periodic        [1] TimePeriod,
        induced         [2] Induced,
        event           [3] ReportingEvent,
        stimulus        [4] OBJECT IDENTIFIER }

-- PossibleError ::= OBJECT IDENTIFIER

ProceduralStatusValue ::= ProceduralStatus (WITH COMPONENT (terminating))

ProviderId ::=    CHOICE {
```

```
            objectReference  [1] ObjectInstance,
            textualName      [2] GraphicString,
            serviceSpecific  [3] ServiceSpecificId,
            unknown          [4] NULL }

        ReportingTriggers ::= SET OF CHOICE {
            periodic         [1] TimePeriod,
            induced          [2] Induced,
            event            [3] ReportingEvent,
            stimulus         [4] OBJECT IDENTIFIER }

        ReportingEvent ::= ENUMERATED {
            registration (0),
            request (1),
            accept (2),
            complete (3),
            corresponding (4),
            bulk (5),
            interruption (6) }

        ServiceSpecificId ::= SEQUENCE {
            service                 OBJECT IDENTIFIER,
            serviceSpecificId       ANY DEFINED BY service }

        UsageDataInfo ::= SEQUENCE {
            accountableObjectReference      [0] ObjectInstance,
            notificationCause               [1] NotificationCause,
            usageInfo                       [2] UsageInfo,
            auditInfo                       [3] AuditInfo                   OPTIONAL,
            dataErrors                      [4] DataErrors,
            providerId                      [5] ProviderId                  OPTIONAL,
            additionalInformation           [6] ManagementExtensionList     OPTIONAL }

--      UsageMeteringControlObjectId ::= SimpleNameType

--      UsageMeteringDataControl ::= SimpleNameType

--
-- Definicoes de outras normas
--
  ControlStatus ::= SET OF ENUMERATED {
    subjectToTest (0) ,
    partOfServicesLocked (1) ,
    reservedForTest (2) ,
    suspended (3)
  }

  ManagementExtension ::= SEQUENCE {
    identifier          OBJECT IDENTIFIER ,
    significance        [1] BOOLEAN  DEFAULT FALSE ,
    information         [2] ANY DEFINED BY  identifier
  }

  ProceduralStatus ::= SET OF ENUMERATED {
    initializationRequired (0) ,
    notInitialized (1) ,
    initializing (2) ,
    reporting (3) ,
    terminating (4)
  }
```

```
UsageInfo ::= SEQUENCE {
        serviceType     OBJECT IDENTIFIER,  -- ServiceType
        usageData       ANY DEFINED BY serviceType }

-- ServiceType ::= OBJECT IDENTIFIER

END
```

11.2 Classes de Objetos Associadas a Recomendação ISO/IEC 10164-10

```
-- ###############################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: mo.dat - 10164-10
                Descrição: Contém as classes de objetos associadas a Recomendação ISO/IEC 10164-10
-- ###############################################################################################

#############################################################
#
#     mo.dat is a database of External MO classes
#     The fields of each record (line) in the file are:
#
# 1) GDMO MO name 2) MO C++ Class Name 3) MO C++ Class Header file
#
#
#############################################################
top                     Top                     Top.h
system                  System                  System.h
systemOIM               System                  SystemOIM.h
log                     Log                     Log.h
eventLogRecord          EventLogRecord          EventLog.h
```

11.3 Classes de Atributos Utilizadas pela Recomendação ISO/IEC 10164-10

```
-- ##########################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: attr.dat - 10164-10
                Descrição: Contém as classes de atributos associadas a Recomendação ISO/IEC 10164-10
-- ##########################################################################################

############################################################
#
#     attr.dat is a database of attribute classes
#     The fields of each record (line) in the file are:
#
# 1) GDMO Attribute Type Name 2) Attribute C++ Class Name 3) C++ Header file
#
#
############################################################
#
# These are the generic attribute types from which other may be derived
# using the DERIVED FROM GDMO clause
#
counter                 Counter                 Counter.h
counterThreshold        CounterThreshold        CounterThld.h
counter-Threshold       CounterThreshold        CounterThld.h
gauge                   Gauge                   Gauge.h
gaugeThreshold          GaugeThreshold          GaugeThld.h
gauge-Threshold         GaugeThreshold          GaugeThld.h
tideMark                TideMark                TideMark.h
tide-Mark               TideMark                TideMark.h
#
# These are commonly used attributes from the DMI document
# (ISO 10165-2 / CCITT X.721) which may be referred to from other GDMO specs
#
objectClass             ObjectClass             ObjectClass.h
managedObjectClass      ObjectClass             ObjectClass.h
managedObjectInstance   ObjectInstance          ObjectInstance.h
eventType               EventType               EventType.h
discriminatorConstruct  MFilter                 Filter.h
destination             DestinationAddressGms   Destination.h
administrativeState     AdministrativeState     AdministrativeState.h
operationalState        OperationalState        OperationalState.h
usageState              UsageState              UsageState.h
availabilityStatus      AvailabilityStatus      AvailabilityStatus.h


#
# The following are attributes used by UMF that were missing
#
recordId                SimpleNameType          SimpleNameType.h

#
# The following are attributes for Usage Metering Function (ISO/IEC 10164-10)
#

accountableObjectReference      ObjectInstance          ObjectInstance.h
```

accountableObjectsReferenceList	ObjectInstanceList	ObjectInstanceList.h
actionResponse	ActionResponse	ActionResponse.h
auditInfo	AuditInfo	AuditInfo.h
controlObjectId	SimpleNameType	SimpleNameType.h
dataObjectId	SimpleNameType	SimpleNameType.h
dataObjectsReferenceList	ObjectInstanceList	ObjectInstanceList.h
dataErrors	DataErrors	DataErrors.h
notificationCause	NotificationCause	NotificationCause.h
providerId	ProviderId	ProviderId.h
reportingTriggers	ReportingTriggers	ReportingTriggers.h
usageInfo	UsageInfo	UsageInfo.h

11.4 Sintaxes de Atributos Associadas a Recomendação ISO/IEC 10164-10

```
-- ###############################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: sintax.dat - 10164-10
                Descrição: Contém as sintaxes de atributos associadas a Recomendação ISO/IEC 10164-10
-- ###############################################################################

#############################################################
#
#     syntax.dat is a database of syntax classes
#     The fields of each record (line) in the file are:
#
# 1) GDMO Syntax Name 2) C++ Class Name 3) C++ Header file
#
#
# INTEGER                       Integer             Integer.h
# REAL                          Real                Real.h
# OCTET STRING                  OctetString         OctetString.h
# IA5String                     String              IA5String.h
# SET OF GraphicString          StringList          StringList.h
# SET OF IA5String              IA5StringList       IA5StringList.h
# UTCTime                       Time                Time.h
# CMISFilter                    MFilter             Filter.h
# Scope                         MScope              Scope.h
# OBJECT IDENTIFIER             ObjId               ObjId.h
# SET OF OBJECT IDENTIFIER      ObjIdList           ObjIdList.h
# ObjectClass                   ObjectClass         ObjectClass.h
# SET OF ObjectClass            ObjectClassList     ObjectClassList.h
#
#############################################################
#
# The following should not be used if the DERIVED FROM clause is used
# for the generic attribute types [ counter, gauge, counter-Threshold
# gauge-Threshold and tide-Mark ] instead of the WITH ATTRIBUTE SYNTAX.
# In this case, the attr.dat file contains the right entries.
# In case though that this is not the case, the syntax of those
# attributes is included below.
#
Count                   Counter             Counter.h
CounterThreshold        CounterThreshold    CounterThld.h
ObservedValue           Gauge               Gauge.h
GaugeThreshold          GaugeThreshold      GaugeThld.h
TideMarkInfo            TideMark            TdeMark.h
#
# The following are OSIMIS-implemented syntaxes that may be re-used
#
SimpleNameType          SimpleNameType      SimpleNameType.h
Boolean                 Boolean             Bool.h
Integer                 Integer             Integer.h
Real                    Real                Real.h
OctetString             OctetString         OctetString.h
GraphicString           GraphicString       GraphicString.h
IA5String               IA5String           IA5String.h
PrintableString         PrintableString     PrintableString.h
```

```
IntegerList                 IntegerList             IntegerList.h
RealList                    RealList                RealList.h
GraphicStringList           GraphicStringList       GraphicStringList.h
IA5StringList               IA5StringList           IA5StringList.h
PrintableStringList         PrintableStringList     PrintableStringList.h
Time                        Time                    Time.h
UTCTime                     Time                    Time.h
GeneralizedTime             Time                    Time.h
CmisScope                   MScope                  Scope.h
CMISScope                   MScope                  Scope.h
Scope                       MScope                  Scope.h
CmisFilter                  MFilter                 Filter.h
CMISFilter                  MFilter                 Filter.h
Filter                      MFilter                 Filter.h
ObjectIdentifier            ObjId                   ObjId.h
ObjectIdentifierList        ObjIdList               ObjIdList.h
ObjectClass                 ObjectClass             ObjectClass.h
ObjectClassList             ObjectClassList         ObjectClassList.h
ObjectInstance              ObjectInstance          ObjectInstance.h
ObjectInstanceList          ObjectInstanceList      ObjectInstanceList.h
DistinguishedName           DName                   DName.h
EventTypeId                 EventType               EventType.h
AttributeId                 AttributeId             AttributeId.h
#
# The following are syntaxes for attributes that may be re-used
#
DiscriminatorConstruct      MFilter                 Filter.h
Destination                 DestinationAddress      Destination.h
AdministrativeState         AdministrativeState     AdministrativeState.h
OperationalState            OperationalState        OperationalState.h
UsageState                  UsageState              UsageState.h
AvailabilityStatus          AvailabilityStatus      AvailabilityStatus.h

#
# The following are syntaxes for Usage Metering Function (ISO/IEC 10164-10)
#

AccountableObjectReference      ObjectInstance      ObjectInstance.h
AccountableObjectsReferenceList ObjectInstanceList  ObjectInstanceList.h
ActionResponse                  ActionResponse      ActionResponse.h
AuditInfo                       AuditInfo           AuditInfo.h
UsageMeteringControlObjectId    SimpleNameType      SimpleNameType.h
UsageMeteringDataObjectId       SimpleNameType      SimpleNameType.h
DataObjectsReferenceList        ObjectInstanceList  ObjectInstanceList.h
DataErrors                      DataErrors          DataErrors.h
NotificationCause               NotificationCause   NotificationCause.h
ProviderId                      ProviderId          ProviderId.h
ReportingTriggers               ReportingTriggers   ReportingTriggers.h
UsageInfo                       UsageInfo           UsageInfo.h
```

11.5 Especificação GDMO da Recomendação M.3100

```
##########################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós-Graduação em Ciência da Computação

                Arquivo: MIB.gdmo - M3100
                Descrição: Contém a especificação em GDMO das classes de objetos associados a
                Recomendação M.3100
##########################################################################################

-- ############ Managed Object templates ###################

network ------------------------------------------------

net MANAGED OBJECT CLASS
     DERIVED FROM "Recommendation X.721:1992":top;
     CHARACTERIZED BY
          userLabelPackage,

          netPackage PACKAGE
          BEHAVIOUR networkDefinition;
               ATTRIBUTES
                         networkId       GET;;;

REGISTERED AS {m3100ObjectClass 1};

managedElement -----------------------------------------

managedElement  MANAGED OBJECT CLASS
          DERIVED FROM "Recommendation X.721:1992":top;
          CHARACTERIZED BY

          createDeleteNotificationsPackage,
          attributeValueChangeNotificationPackage,
          stateChangeNotificationPackage,
--        audibleVisualLocalAlarmPackage,
--        resetAudibleAlarmPackage,
          userLabelPackage,
          vendorNamePackage,
          versionPackage,
          locationNamePackage,
          currentProblemListPackage,
          externalTimePackage,
--        systemTimingSourcePackage,

managedElementPackage PACKAGE
          BEHAVIOUR managedElementBehaviour;
          ATTRIBUTES
           managedElementId       GET,
--         systemTitle            GET-REPLACE,
--         alarmStatus            GET,
           administrativeState    GET-REPLACE,
           operationalStateGET,
           usageState             GET;;;
--        NOTIFICATIONS
           --"Recommendation X.721:1992":environmentAlarm,
```

```
        --"Recommendation X.721:1992":equipamentAlarm,
        --"Recommendation X.721:1992":communicationsAlarm,
        --"Recommendation X.721:1992":processingErrorAlarm;;;

REGISTERED AS {m3100ObjectClass 3};

managedElementR1 ----------------------------------------

managedElementR1        MANAGED OBJECT CLASS
        DERIVED FROM        managedElement;
        CHARACTERIZED BY

--      alarmSeverityAssignmentPointerPackage,

managedElementR1Package         PACKAGE;;
        -- NOTIFICATIONS
                -- "Recommendation X.721:1992":environmentalAlarm
                --      "Recommendation Q.821:1992":logRecordIdParameter
                --      "Recommendation Q.821:1992":correlatedRecordNameParameter
                --      "Recommendation Q.821:1992":suspectObjectListParameter,
                -- "Recommendation X.721:1992":equipmentAlarm
                --      "Recommendation Q.821:1992":logRecordIdParameter
                --      "Recommendation Q.821:1992":correlatedRecordNameParameter
                --      "Recommendation Q.821:1992":suspectObjectListParameter,
                -- "Recommendation X.721:1992":communicationsAlarm
                --      "Recommendation Q.821:1992":logRecordIdParameter
                --      "Recommendation Q.821:1992":correlatedRecordNameParameter
                --      "Recommendation Q.821:1992":suspectObjectListParameter,
                -- "Recommendation X.721:1992":processingErrorAlarm
                --      "Recommendation Q.821:1992":logRecordIdParameter
                --      "Recommendation Q.821:1992":correlatedRecordNameParameter
                --      "Recommendation Q.821:1992":suspectObjectListParameter;;;

REGISTERED AS {m3100ObjectClass 27};

-- terminationPoint ----------------------------------------

terminationPoint      MANAGED OBJECT CLASS
    DERIVED FROM   "Recommendation X.721:1992":top;
    CHARACTERIZED BY

        createDeleteNotificationsPackage,
    attributeValueChangeNotificationPackage,
        stateChangeNotificationPackage,
        operationalStatePackage,
--      crossConnectionPointerPackage,
        characteristicInformationPackage,
        networkLevelPackage,
        tmnCommunicationsAlarmInformationPackage,
--      alarmSeverityAssignmentPointerPackage,

        terminationPointPackage PACKAGE
    BEHAVIOUR terminationPointBehaviour;
    ATTRIBUTES

-- terminationPointId was added by BOB & Ariadne at 01/27/97. Testing only.
        terminationPointId      GET,
        supportedByObjectList   GET;;;

REGISTERED AS {m3100ObjectClass 8};
```

```
-- connectionTerminationPointSink ----------------------------------

connectionTerminationPointSink MANAGED OBJECT CLASS
     DERIVED FROM   terminationPoint;
     CHARACTERIZED BY
         ctpInstancePackage,
         channelNumberPackage,

         connectionTerminationPointSinkPackage   PACKAGE
     BEHAVIOUR connectionTerminationPointSinkBehaviour;
     ATTRIBUTES
          downstreamConnectivityPointer  PERMITTED VALUES

   ASN1DefinedTypesModule.CTPDownstreamPointer    GET SET-BY-CREATE;;;

REGISTERED AS {m3100ObjectClass 6};

-- connectionTerminationPointSource --------------------------------

connectionTerminationPointSource      MANAGED OBJECT CLASS
     DERIVED FROM   terminationPoint;
     CHARACTERIZED BY
         ctpInstancePackage,
         channelNumberPackage,

         connectionTerminationPointSourcePackage PACKAGE
     BEHAVIOUR connectionTerminationPointSourceBehaviour;;;
--     ATTRIBUTES
--          upstreamConnectivityPointer   PERMITTED VALUES;;;
--               CTPDownstreamPointer ASN1DefinedTypesModule.CTPUpstreamPointer     GET SET-BY-
CREATE;;;

REGISTERED AS {m3100ObjectClass 7};

-- connectionTerminationPointBidirectional -------------------------

connectionTerminationPointBidirectional MANAGED OBJECT CLASS
     DERIVED FROM   connectionTerminationPointSource;
          -- connectionTerminationPointSink;

REGISTERED AS {m3100ObjectClass 5};

-- trailTerminationPointSink ---------------------------------

trailTerminationPointSink     MANAGED OBJECT CLASS
     DERIVED FROM   terminationPoint;
     CHARACTERIZED BY
         administrativeStatePackage,
         supportableClientListPackage,
         ttpInstancePackage,
         operationalStatePackage,
     trailTerminationPointSinkPackage PACKAGE
     BEHAVIOUR trailTerminationPointSinkBehaviour;

     ATTRIBUTES
          upstreamConnectivityPointer    GET SET-BY-CREATE;;;

REGISTERED AS {m3100ObjectClass 10};

-- trailTerminationPointSource -------------------------------
```

```
trailTerminationPointSource    MANAGED OBJECT CLASS
    DERIVED FROM    terminationPoint;
    CHARACTERIZED BY
        administrativeStatePackage,
        supportableClientListPackage,
        ttpInstancePackage,
        operationalStatePackage,
    trailTerminationPointSourcePackage PACKAGE
    BEHAVIOUR trailTerminationPointSourceBehaviour;
    ATTRIBUTES
        downstreamConnectivityPointer    GET SET-BY-CREATE;;;

REGISTERED AS {m3100ObjectClass 11};

trailTerminationPointBidirectional    MANAGED OBJECT CLASS
    DERIVED FROM    trailTerminationPointSource;
--              trailTerminationPointSink;
    CHARACTERIZED BY
trailTerminationPointBidirectionalPackage PACKAGE
    BEHAVIOUR trailTerminationPointBidirectionalBehaviour;;;

REGISTERED AS {m3100ObjectClass 9};


-- ############ Packages templates ##########################

administrativeStatePackage PACKAGE
        ATTRIBUTES
                administrativeState        GET-REPLACE;
REGISTERED AS {m3100Package 1};

alarmSeverityAssignmentPointerPackage PACKAGE
        ATTRIBUTES
                alarmSeverityAssignmentProfilePointer GET-REPLACE;
REGISTERED AS {m3100Package 3};

attributeValueChangeNotificationPackage   PACKAGE
        NOTIFICATIONS
                attributeValueChange;
REGISTERED AS {m3100Package 4};

audibleVisualLocalAlarmPackage PACKAGE
        ACTIONS
                allowAudibleVisualLocalAlarm,
                inhibitAudibleVisualLocalAlarm;
REGISTERED AS {m3100Package 5};

channelNumberPackage PACKAGE
        ATTRIBUTES
                channelNumber  GET;
REGISTERED AS {m3100Package 6};

characteristicInformationPackage PACKAGE
        ATTRIBUTES
                characteristicInformation  GET;
REGISTERED AS {m3100Package 7};

crossConnectionPointerPackage PACKAGE
        ATTRIBUTES
                crossConnectionObjectPointer      GET;
REGISTERED AS {m3100Package 11};
```

```
ctpInstancePackage PACKAGE
        ATTRIBUTES
                cTPId   GET SET-BY-CREATE;
REGISTERED AS {m3100Package 12};

createDeleteNotificationsPackage PACKAGE
        NOTIFICATIONS
                objectCreation,
                objectDeletion;
REGISTERED AS {m3100Package 10};

currentProblemListPackage PACKAGE
        ATTRIBUTES
                currentProblemList      GET;
REGISTERED AS {m3100Package 13};

externalTimePackage PACKAGE
        ATTRIBUTES
                externalTime    GET-REPLACE;
REGISTERED AS {m3100Package 16};

locationNamePackage PACKAGE
        ATTRIBUTES
                locationName    GET-REPLACE;
REGISTERED AS {m3100Package 17};

networkLevelPackage PACKAGE
        ATTRIBUTES
                networkLevelPointer     GET-REPLACE;
REGISTERED AS {m3100Package 18};

operationalStatePackage PACKAGE
        ATTRIBUTES
                operationalState  GET;
REGISTERED AS {m3100Package 19};

--resetAudibleAlarmPackage PACKAGE
--      ACTIONS
--              "Recommendation Q.821: 1992":resetAudibleAlarm;
--REGISTERED AS {m3100Package 23};

stateChangeNotificationPackage    PACKAGE
        NOTIFICATIONS
                stateChange;
REGISTERED AS {m3100Package 28};

supportableClientListPackage PACKAGE
        ATTRIBUTES
                supportableClientList     GET;
REGISTERED AS {m3100Package 27};

systemTimingSourcePackage PACKAGE
        ATTRIBUTES
                systemTimingSource      GET-REPLACE;
REGISTERED AS {m3100Package 29};

tmnCommunicationsAlarmInformationPackage PACKAGE
        ATTRIBUTES
--              alarmStatus             GET;
                currentProblemList      GET;
```

```
--      NOTIFICATIONS
--              "Recommentdation X.721: 1992":communicationAlarm
--              "Recommentdation Q.821: 1992":logRecordIdParameter
--              "Recommentdation Q.821: 1992":correlatedRecordNameParameter
--              "Recommentdation Q.821: 1992":suspectObjectListParameter;
REGISTERED AS {m3100Package 30};

ttpInstancePackage PACKAGE
        ATTRIBUTES
                tTPId                   GET SET-BY-CREATE;
REGISTERED AS {m3100Package 31};

userLabelPackage PACKAGE
        ATTRIBUTES
                userLabel               GET-REPLACE;
REGISTERED AS {m3100Package 32};

vendorNamePackage PACKAGE
        ATTRIBUTES
                vendorName              GET-REPLACE;
REGISTERED AS {m3100Package 33};

versionPackage PACKAGE
        ATTRIBUTES
                version                 GET-REPLACE;
REGISTERED AS {m3100Package 34};


-- ############ Attributes templates ##########################

alarmSeverityAssignmentProfilePointer ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.PointerOrNull;
        MATCHES FOR EQUALITY;
        BEHAVIOUR alarmSeverityAssignmentPointerBehaviour;
REGISTERED AS {m3100Attribute 5};

alarmStatus ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.AlarmStatus;
        MATCHES FOR EQUALITY;
        BEHAVIOUR alarmStatusBehaviour;
REGISTERED AS {m3100Attribute 6};

channelNumber ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.ChannelNumber;
        MATCHES FOR EQUALITY, ORDERING;
REGISTERED AS {m3100Attribute 7};

characteristicInformation ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.CharacteristicInformation;
        MATCHES FOR EQUALITY;
        BEHAVIOUR characteristicInformationBehaviour;
REGISTERED AS {m3100Attribute 8};

cTPId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCL-.SimpleNameType;
        MATCHES FOR EQUALITY;
REGISTERED AS {m3100Attribute 13};

crossConnectionObjectPointer ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.CrossConnectionObjectPointer;
--      WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.MultipleConnections;
```

```
--      WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.SimpleNameType;
--      WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.ObjectInstance;
        MATCHES FOR EQUALITY;
        BEHAVIOUR crossConnectionObjectPointerBehaviour;
REGISTERED AS {m3100Attribute 16};

currentProblemList ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.CurrentProblemList;
        BEHAVIOUR currentProblemListBehaviour;
REGISTERED AS {m3100Attribute 17};

downstreamConnectivityPointer ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.DownstreamConnectivityPointer;
        MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
        BEHAVIOUR downstreamConnectivityPointerBehaviour;
REGISTERED AS {m3100Attribute 19};

externalTime ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.ExternalTime;
        MATCHES FOR EQUALITY;
        BEHAVIOUR externalTimeBehaviour;
REGISTERED AS {m3100Attribute 21};

locationName ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.LocationName;
        MATCHES FOR EQUALITY, SUBSTRINGS;
        BEHAVIOUR locationNameBehaviour;
REGISTERED AS {m3100Attribute 27};

managedElementId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.SimpleNameType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR managedElementIdBehaviour;
REGISTERED AS {m3100Attribute 28};

networkId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.SimpleNameType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR networkIdBehaviour;
REGISTERED AS {m3100Attribute 29};

networkLevelPointer ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.ObjectInstance;
        MATCHES FOR EQUALITY;
REGISTERED AS {m3100Attribute 31};

supportedByObjectList ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.ObjectList;
        MATCHES FOR EQUALITY, SET-COMPARISON, SET-INTERSECTION;
        BEHAVIOUR supportedByObjectListBehaviour;
REGISTERED AS {m3100Attribute 40};

systemTimingSource ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.SystemTimingSource;
        MATCHES FOR EQUALITY;
        BEHAVIOUR systemTimingSourceBehaviour;
REGISTERED AS {m3100Attribute 41};

upstreamConnectivityPointer ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.ConnectivityPointer;
        MATCHES FOR EQUALITY;
```

```
        BEHAVIOUR upstreamConnectivityPointerBehaviour;
REGISTERED AS {m3100Attribute 49};

userLabel ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.UserLabel;
         MATCHES FOR EQUALITY, SUBSTRINGS;
        BEHAVIOUR userLabelBehaviour;
REGISTERED AS {m3100Attribute 50};

vendorName ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.VendorName;
         MATCHES FOR EQUALITY, SUBSTRINGS;
        BEHAVIOUR vendorNameBehaviour;
REGISTERED AS {m3100Attribute 51};

version ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.Version;
        MATCHES FOR EQUALITY, SUBSTRINGS;
        BEHAVIOUR versionBehaviour;
REGISTERED AS {m3100Attribute 52};

-- terminationPointId was added by BOB & Ariadne at 01/27/97. Testing only.
terminationPointId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.SimpleNameType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR terminationPointIdBehaviour;
REGISTERED AS {m3100Attribute 53};

-- ############ Notifications and actions templates ##########################

allowAudibleVisualLocalAlarm ACTION
        BEHAVIOUR allowAudibleVisualLocalAlarmBehaviour;
REGISTERED AS {m3100Action 3};

inhibitAudibleVisualLocalAlarm ACTION
        BEHAVIOUR inhibitAudibleVisualLocalAlarmBehaviour;
REGISTERED AS {m3100Action 6};

-- Imported Notifications

objectCreation NOTIFICATION
  BEHAVIOUR             objectCreationBehaviour;
  WITH INFORMATION SYNTAX   Notification-ASN1Module.ObjectInfo
  AND ATTRIBUTE IDS
    sourceIndicator     sourceIndicator,
    attributeList       attributeList;
  REGISTERED AS         { smi2Notification 6 };


objectDeletion NOTIFICATION
  BEHAVIOUR             objectDeletionBehaviour;
  WITH INFORMATION SYNTAX   Notification-ASN1Module.ObjectInfo
  AND ATTRIBUTE IDS
    sourceIndicator     sourceIndicator,
    attributeList       attributeList;
  REGISTERED AS         { smi2Notification 7 };


attributeValueChange NOTIFICATION
  BEHAVIOUR             attributeValueChangeBehaviour;
  WITH INFORMATION SYNTAX   Notification-ASN1Module.AttrValChangeInfo
```

```
  AND ATTRIBUTE IDS
    sourceIndicator       sourceIndicator,
    attrValChangeDefinition attrValChangeDefinition;
  REGISTERED AS           { smi2Notification 1 };


stateChange NOTIFICATION
  BEHAVIOUR               stateChangeBehaviour;
  WITH INFORMATION SYNTAX   Notification-ASN1Module.AttrValChangeInfo
  AND ATTRIBUTE IDS
    sourceIndicator       sourceIndicator,
    stateChangeDefinition  stateChangeDefinition;
  REGISTERED AS           { smi2Notification 14 };


-- ############## Name Binding Templates ##############

net-system NAME BINDING
        SUBORDINATE OBJECT CLASS          net AND SUBCLASSES;
        NAMED BY
        SUPERIOR OBJECT CLASS             system AND SUBCLASSES;
        WITH ATTRIBUTE                    networkId;
        CREATE      WITH-AUTOMATIC-INSTANCE-NAMING;
    DELETE      DELETES-CONTAINED-OBJECTS;

REGISTERED AS {m3100NameBinding 16};

terminationPoint-system NAME BINDING
        SUBORDINATE OBJECT CLASS          terminationPoint AND SUBCLASSES;
        NAMED BY
        SUPERIOR OBJECT CLASS             system AND SUBCLASSES;
        WITH ATTRIBUTE                    terminationPointId;
        CREATE      WITH-AUTOMATIC-INSTANCE-NAMING;
    DELETE      DELETES-CONTAINED-OBJECTS;

REGISTERED AS {m3100NameBinding 25};

--net-net NAME BINDING
--      SUBORDINATE OBJECT CLASS          net AND SUBCLASSES;
--      NAMED BY
--      SUPERIOR OBJECT CLASS             net AND SUBCLASSES;
--      WITH ATTRIBUTE                    networkId;
--      BEHAVIOUR                 networkCreateBehaviour;
-- REGISTERED AS {m3100NameBinding 17};

managedElement-net NAME BINDING
        SUBORDINATE OBJECT CLASS          managedElement;
        NAMED BY
        SUPERIOR OBJECT CLASS             net AND SUBCLASSES;
        WITH ATTRIBUTE                    managedElementId;
        BEHAVIOUR                 managedElementCreateBehaviour;
REGISTERED AS {m3100NameBinding 15};


-- ############ Behaviours templates ##########################

alarmSeverityAssignmentPointerBehaviour BEHAVIOUR
DEFINED AS "This attribute identifies an Alarm Severity Assignment Profile object.";

---
alarmSeverityAssignmentPointerPackageBehaviour BEHAVIOUR
```

DEFINED AS "If the alarm severity assignment profile pointer is NULL, then one of the following two choices when reporting alarms:
agent assigns the severity or
the value 'indeterminated' is user.";

---
alarmStatusBehaviour BEHAVIOUR
DEFINED AS "The Alarm Status attribute type indicates the accurrence of an abnormal condition relating to an object. This attribute may also function as a summary indicator of alarm conditions associated wioth a specific resource. It is used to indicate the existence of an alarm condition, a pending alarm condition such as threshold situations, or (when used as a summary indicator) the highest severity of active alarm conditions. When used as a summary indicator, the order of severity (from highest to lowest) is:
activeReportable-Critical;
activeReportable-Major;
activeReportable-Minor;
activeReportable-Indeterminate;
activeReportable-Warning;
activePending and
cleared.";

---
characteristicInformationBehaviour BEHAVIOUR
DEFINED AS "The value of this attribute is used to verify the connectability of instances of the terminaion point sub-classes.";

---
connectionTerminationPointSinkBehaviour BEHAVIOUR
DEFINED AS "This managed object terminates a link connection. The downstream connectivity pointer attribute points to the termination point managed object, within the same managed element, that receives information (traffic) from thios termination point at the same layer, or is null. The referenced ebject shall be an instance of one of the following classes or its sub-classes: Trail Termination Point Sink, Trail Termination Point Biderectional, Connection Termination Point Source, Connection Termination Point Bidirectional. The downstream conectivity pointer may identify one or more objects depending on whether the signal is connected to one or more termination point objects.";

---
connectionTerminationPointSourceBehaviour BEHAVIOUR
DEFINED AS "This managed object originates a link connection. The upstream connectivity pointer attribute points to the termination point managed object, within the same managed element, that sends information (traffic) to this termination point at the same layer, or is null. The referenced object shall be an instance of one of the following classes or its sub-classes: Trail Termination Point Source, Trail Termination Point Biderectional, Connection Termination Point Sink, Connection Termination Point Bidirectional.";

---
crossConnectionObjectPointerBehaviour BEHAVIOUR
DEFINED AS "This attribute points to a managed abject such as a Cross-connection, a GTP or a Fabric. When a termination point is neither connected nor reserved for connections. its CrossConnectionObjectPointer points to the Fabric Object responsible for its connection.";

---
cTPSink-TTPBehaviour BEHAVIOUR
DEFINED AS "The name binding represents a relationship in which a TTP sends information (traffic) to a sink CTP. When automatic instance naming is used, the choice of name binding is left as a local matter.";
---
cTPSource-TTPBehaviour BEHAVIOUR
DEFINED AS "The same binding represents a relationship in qhich a TTP receives information (traffic) from a source CTP.

When automatic instance naming is used, the choice of name binding is left as a local matter.";

---

currentProblemListBehaviour BEHAVIOUR
DEFINED AS "The Current Problem List attribute type identifies the current existing problems, with severity, associated with the managed object.";

---
downstreamConnectivityPointerBehaviour BEHAVIOUR
DEFINED AS "The matching for equality is applicable for all choices of the syntax. The set operations are permitted only when the choice of the syntax is correspond to either broadcast or concatenated broadcast.";

---
externalTimeBehaviour BEHAVIOUR
DEFINED AS "The External time attribute provides time-of-day system time. The attribute functions as a reference for all time stamp activities in the managed element.";

---
locationNameBehaviour BEHAVIOUR
DEFINED AS "The Location Name attribute type identifies a location.";

---
managedElementBehaviour BEHAVIOUR
DEFINED AS " The Managed Element object class is a class of managed objects represents telecommunications equipment or TMN entities (either groups or parts) within the telecommunications network that performs managed element functions, i.e., provides support and/or service to the subscriber. Managed elements may or may not additionally perform mediation/OS functions. A managed elemenet communicates with the manager over one or more standard Q-interfaces for purpose of being monitores and/or controlled. A managed element contains equipament that may or may not be geographycally distributed.

When the attribute value change notification package is present, the attributeValueChange notification defined iun Recommendation X.721 shall be emitted when the value of one of tje following attributes changes: alarm status, user label, version, location name and current problem list. For the above attributes that are in conditional packages, the behaviour for emitting the attribute value change notification applies only when the corresponding conditional packages are present in the managed abject. When the state change notification package is present, the stateChangeNotification defined in Recommendation X.721 shall be emitted if the value of the administrative state or operational state or usage state changes.";

---
managedElementCreateBehaviour BEHAVIOUR
DEFINED AS "Managed Element object is not created or deleted by system management protocol. The object is created when initializing the managed element.";

---
managedElementIdBehaviour BEHAVIOUR
DEFINED AS "The Managed Element Id is an attribute type whose distinguished value can be used as a RDN when naming an instance of the Managed Element object class.";

---
networkCreateBehaviour BEHAVIOUR
DEFINED AS "Network object is not created or deleted by system management protocol. The object is created when initializing the network.";

---
networkDefinition BEHAVIOUR
DEFINED AS "This Equipment object class is a class of managed objects that represents physical components of a managed element, including replaceable3 components. An instance of this object is present in a single geographic location. An equipament may be nested within another equipament, thereby creating a containment relationship. The equipment type shall be identified by sub-classing this object class. Either the name of the sub-class or an attribute may be used for identifyine the equipment type.

When an attribute value change notification package is present, the attributeValueChange notification defined in Recommendation X.721 shall be emitted when the value of one of the following attributes changes: alarm status, affected object list, user label, version, location name and current problem list. Because the above attributes are

all in conditional packages, the behaviour for emiting the attribute value change notification applies only when the corresponding conditional packages are present in the managed object. When the state changes notification package is present, the stateChangeNotification defined in Recommendation X.721 shall be emitted if the value of administrative state or operational states changes (when the administrativeOperationalStates conditional package is present).";

---
networkIdBehaviour BEHAVIOUR
DEFINED AS " The Network Id is an attribute type whose distinguished can be used as an RDN when naming an instance of the Network object class.";

---
networkLevelPackageBehaviour BEHAVIOUR
DEFINED AS "The network level pointer identifies a network level object. The value of the network level pointer shall only be modified by the managing system.";

---
supportableClientListBehaviour BEHAVIOUR
DEFINED AS "The value of this attribute is the list of object classes representing the clients which the particular managed object is capable of supporting. This may be a subset of the client layers identified in Recommendation G.803 by the particular server layer managed object.";

---
supportedByObjectListBehaviour BEHAVIOUR
DEFINED AS "The Supported By List is an attribute type whose the value identifies a set of object instances wich are capable of directly affecting a given managed object. The object instances include both physical and logical objects. This attribute does not force internal details to be specified, but only the necessary level of detail required for managment. If the object instances supporting the managed object are unknown to that object, then this attribute is an empty set.";

---
systemTimingSourceBehaviour BEHAVIOUR
DEFINED AS "The System Timing Source attribute is used to specify the primary and secondary managed element timing sourc for synchronization.";

---
terminationPointBehaviour BEHAVIOUR
DEFINED AS "This managed object represents the termnination of a transport entity, such as a trail or a cionnection. The characteristic information attribute is used to identify equivalence between sub-classes of termination point in order to determine whether cross connection or connectivity is possible. The operational state reflects the perceived ability to generate and/or receive a valid signal. Sub-classes of termination point shall specify the attributes and states for wich attributes value change and state change notifications will be generated.";

---
tmnCommunicationsAlarmInformationBehaviour BEHAVIOUR
DEFINED AS "An alarm report wich contains a Perceived Severity parameter with a value of 'cleared' and the Correlated Notifications parameter shall only indicate the clearing of those alarms whose Notifications Identifiers are included in the set of Correlated Notifications. An alarm report wich contains a Perceived Severity parameter with the value 'cleared', but no Correlated Notifications parameter, shall indicate the clearing of alarms based on the value of the ALarm Type, Probable Cause and Specific Problems parameters.

The parameters that are associated with the communications alarm, if present, are placed in individual elements of the SET OF ManagementExtension in the additionalInformation field of the notification.";

---
trailTerminationPointBidirectionalBehaviour BEHAVIOUR
DEFINED AS "The operational state is disabled if either the sink or source part of the termination point is disabled.";

--

trailTerminationPointNameBindingBehaviour BEHAVIOUR
DEFINED AS "When automatic instance naming is used, the choice of name binding is left as a local matter.";

---

trailTerminationPointSinkBehaviour BEHAVIOUR
DEFINED AS "This managed object represents a termination point where a trail is terminated. It represents the access point in a layer network wich is a focus for both the trail relationship and the client/server relatioonship.

The operational state reflects the perceived ability to receive a valid signal. If the termiantion pointz detected that a signal received has failed or is unable to process the incoming signal, then the operational state will have the value disabled.

When the admnistrative state is locked, the termination point is administratively removed from service. When the administrative state is unlocked, the termination point is administratively in service. Changes to the administrative state have no effect on the connectivity pointer.

A change in the operational state shall cause a state change nctification. If administrative state is present in an instance of trail termination point sink class, it shall not emit a state change notification. However, sub-classes of trail termiantion point sink may modify this behaviour to require this notification. Sub-classes of trail termination point sink shall specify the attribute for wich attribute value change notification should be generated.

The upstream connectivity pointer attribute points to the termination point managed object, within the same managed element, that sends information (traffic) to this termination point at the same layer, or is null. The referenced object shall be an instance of one of the following classes or its sub-classes: Connection Point Sink or Bidirectional (single or a concatenated sequence) or Trail Termination Point Source or Bidirectional.";

---

trailTerminationPointSourceBehaviour BEHAVIOUR
DEFINED AS "This managed object represents a termination point where a trail is originated. It represents the acess point in a layer network which is a focus for both the trail relationship and the client/server relationship.

The operational state reflects the perceived ability to generate a valid signal. If the termination point detects that a valid signal can be generated, then the state will have the value disabled.

When the admnistrative state is locked, the termination point is administratively removed from service. When the administrative state is unlocked, the termination point is administratively in service. Changes to the administrative state have no effect on the connectivity pointer.

A change in the operational state shall cause a state change notification. If administrative state is present in an instance of trail termination point source class, it shall not emit a state change notification. However, sub-classes of trail termiantion point source may modify this behaviour to require this notification. Sub-classes of trail termination point source shall specify the attribute for wich attribute value change notification should be generated.

The downstream connectivity pointer attribute points to the termination point managed object, within the same managed element, that sends information (traffic) to this termination point at the same layer, or is null. The referenced object shall be an instance of one of the following classes or its sub-classes: Connection Point Source or Bidirectional (single or a concatenated sequence) or Trail Termination Point Sink or Bidirectional (single or a set if connected to more than one trail termiantion point sink objects).";

---

tTPIdBehaviour BEHAVIOUR
DEFINED AS "The Trail Termination Point Id is an attribute type whose distinguished value can be used as an RDN when naming an instance of the Trail TErmination Point object class.";

---

upstreamConnectivityPointerBehaviour BEHAVIOUR
DEFINED AS "The matching for equality is applicable for all the choices of the syntax.";

---

userLabelBehaviour BEHAVIOUR

```
DEFINED AS "The User Label attribute type asigns a user friendly name to the associated object.";

---
vendorNameBehaviour BEHAVIOUR
DEFINED AS "The Vendor Name attribute identifies the vendor pf the associated managed object.";

---
versionBehaviour BEHAVIOUR
DEFINED AS "The Version attribute type identifies the version of the associated object.";

-- BOB & Ariadne at 01/27/97. Testing only.
terminationPointIBehaviour BEHAVIOUR
DEFINED AS "Questo maledeto atributo.";
```

11.6 Especificação ASN.1 das Sintaxes de Atributos da Recomendação M.3100

```
-- ##########################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: ASN1DefinedTypesModule.py - M3100
                Descrição: Contém a especificação em ASN1 das sintaxes de atributos associados a
                Recomendação M.3100
-- ##########################################################################################

ASN1DefinedTypesModule {informationMode(0) asn1Modules(2) asn1DefinedTypesModules(0)}

DEFINITIONS IMPLICIT TAGS ::=

%{
#include <stdio.h>
#ifndef  ICR1
#include <isode/isoaddrs.h>
#include <isode/pepsy/ACS-types.h>
#else
#include <isode/ll/isoaddrs.h>
#include <isode/asn1/ACS-types.h>
#endif
%}


PREFIXES build parse print

BEGIN

IMPORTS
        PSAPaddr FROM DSE
        AE-title FROM ACS
        ObjectClass, ObjectInstance, Scope FROM CMIP
        AdministrativeState, GraphicString, SimpleNameType FROM SMI {
                joint-iso-ccitt
                ms(9)
                part2(2)
                asn1Module(2)
                1
        }
        DistinguishedName FROM IF;


SECTIONS build parse print

--supporting productions

AlarmStatus ::= ENUMERATED {
        cleared (0),
        activeReportable-Indeterminate (1),
        activeReportable-Warning (2),
        activeReportable-Minor (3),
        activeReportable-Major (4),
        activeReportable-Critical (5),
        activePending (6)}
```

```
ProbableCause ::= CHOICE {
        globalValue OBJECT IDENTIFIER ,
        localValue INTEGER
        }

CurrentProblem ::= SEQUENCE {
        problem    [0] ProbableCause,
        alarmStatus [1] AlarmStatus
        }

CurrentProblemList ::= SET OF CurrentProblem

ProblemCause ::= CHOICE {
        unknown NULL,
        integerValue INTEGER
        }

SupportableClientList ::= SET OF ObjectClass

SystemTiming ::= SEQUENCE {
        sourceType ENUMERATED {
                        internalTimingSource (0),
                        remoteTimingSource (1),
                        slavedTimingTerminationSignal (2)
                        },
        sourceID ObjectInstance
        }

SystemTimingSource ::= SEQUENCE {
        primaryTimingSource SystemTiming,
        secondaryTimingSource SystemTiming
        }


SystemTitle ::= CHOICE {
        distinguishedName DistinguishedName,
        oid OBJECT IDENTIFIER ,
        nothing NULL
        }


ObjectList ::= SEQUENCE OF ObjectInstance


MultipleConnections ::= SET OF CHOICE {
        downstreamNotConnected          ObjectInstance,
        downstreamConnected             ObjectInstance,
        upstreamNotConnected            ObjectInstance,
        upstreamConnected               ObjectInstance
        }


ConnectivityPointer ::= CHOICE {
        none    NULL,
        single  ObjectInstance,
        concatenated ObjectList
        }
```

```
CrossConnectionObjectPointer ::= CHOICE {
        notConnected            ObjectInstance,
        connected               ObjectInstance,
        multipleConnections     MultipleConnections
        }

DownstreamConnectivityPointer ::= CHOICE {
        none    NULL,
        single  ObjectInstance,
        concatenated    ObjectList,
        broadcast       ObjectList,
        broadcastConcatenated[1] ObjectList
        }


PointerOrNull ::= CHOICE {
        pointer ObjectInstance,
        null    NULL
        }


CharacteristicInformation ::= OBJECT IDENTIFIER

END
```

11.7 Classes de Objetos Associadas a Recomendação M.3100

```
--#############################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: mo.dat - M3100
                Descrição: Contém as classes de objetos associadas a Recomendação M.3100
--#############################################################################################

#############################################################
#
#     mo.dat is a database of External MO classes
#     The fields of each record (line) in the file are:
#
# 1) GDMO MO name 2) MO C++ Class Name 3) MO C++ Class Header file
#
#
#############################################################
top             Top         Top.h
system          System       System.h
```

11.8 Classes de Atributos Utilizadas pela Recomendação M.3100

```
--#############################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: Attr.dat - M3100
                Descrição: Contém as classes de atributos associadas a Recomendação M.3100
--#############################################################################################

#############################################################
#
#     attr.dat is a database of attribute classes
#     The fields of each record (line) in the file are:
#
# 1) GDMO Attribute Type Name 2) Attribute C++ Class Name 3) C++ Header file
#
#
#############################################################
#
# These are commonly used attributes from the DMI document
# (ISO 10165-2 / CCITT X.721) which may be referred to from other GDMO specs
#
objectClass                 ObjectClass              ObjectClass.h
managedObjectClass          ObjectClass              ObjectClass.h
managedObjectInstance       ObjectInstance           ObjectInstance.h
eventType                   EventType                EventType.h
discriminatorConstruct      MFilter                  Filter.h
destination                 DestinationAddressGms    Destination.h
administrativeState         AdministrativeState      AdministrativeState.h
operationalState            OperationalState         OperationalState.h
usageState                  UsageState               UsageState.h
availabilityStatus          AvailabilityStatus       AvailabilityStatus.h
systemTitle                 SystemTitle              SystemTitle.h

#
# These are the generic attribute types from which other may be derived
# using the DERIVED FROM GDMO clause
#
alarmSeverityAssignmentProfilePointer   PointerOrNUll              PointerOrNull.h
alarmStatus                             AlarmStatus                AlarmStatus.h
channelNumber                           ChannelNumber              ChannelNumber.h
characteristicInformation               CharacteristicInformation  CharacteristicInformation.h
cTPId                                   SimpleNameType             SimpleNameType.h
crossConnectionObjectPointer            CrossConnectionObjectPointer CrossConnectionObjectPointer.h
currentProblemList                      CurrentProblemList         CurrentProblemList.h
downstreamConnectivityPointer           DownstreamConnectivityPointer DownstreamConnectivityPointer.h
externalTime                            ExternalTime               ExternalTime.h
locationName                            LocationName               LocationName.h
managedElementId                        SimpleNameType             SimpleNameType.h
networkId                               SimpleNameType             SimpleNameType.h
networkLevelPointer                     NetworkLevelPointer        NetworkLevelPointer.h
supportableClientList                   SupportableClientList      SupportableClientList.h
supportedByObjectList                   SupportedByObjectList      SupportedByObjectList.h
systemTimingSource                      SystemTimingSource         SystemTimingSource.h
tTPId                                   SimpleNameType             SimpleNameType.h
upstreamConectivityPointer              ConectivityPointer         ConectivityPointer.h
userLabel                               UserLabel                  UserLabel.h
```

vendorName	VendorName	VendorName.h
version	Version	Version.h

11.9 Sintaxes de atributo associadas a recomendação M.3100

```
--##############################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: Syntax.dat - M3100
                Descrição: Contém as sintaxes dos atributos associadas a Recomendação M.3100
--
##############################################################################################
#
#
#     syntax.dat is a database of syntax classes
#     The fields of each record (line) in the file are:
#
# 1) GDMO Syntax Name 2) C++ Class Name 3) C++ Header file
#
#
# INTEGER                       Integer             Integer.h
# REAL                          Real                Real.h
# OCTET STRING                  OctetString         OctetString.h
# IA5String                     String              IA5String.h
# SET OF GraphicString          StringList          StringList.h
# SET OF IA5String              IA5StringList       IA5StringList.h
# UTCTime                       Time                Time.h
# CMISFilter                    MFilter             Filter.h
# Scope                         MScope              Scope.h
# OBJECT IDENTIFIER             ObjId               ObjId.h
# SET OF OBJECT IDENTIFIER      ObjIdList           ObjIdList.h
# ObjectClass                   ObjectClass         ObjectClass.h
# SET OF ObjectClass            ObjectClassList     ObjectClassList.h
#
############################################################
#
# The following should not be used if the DERIVED FROM clause is used
# for the generic attribute types [ counter, gauge, counter-Threshold
# gauge-Threshold and tide-Mark ] instead of the WITH ATTRIBUTE SYNTAX.
# In this case, the attr.dat file contains the right entries.
# In case though that this is not the case, the syntax of those
# attributes is included below.
#
Count                   Counter             Counter.h
CounterThreshold        CounterThreshold    CounterThld.h
ObservedValue           Gauge               Gauge.h
GaugeThreshold          GaugeThreshold      GaugeThld.h
TideMarkInfo            TideMark            TdeMark.h
#
# The following are OSIMIS-implemented syntaxes that may be re-used
#
SimpleNameType          SimpleNameType      SimpleNameType.h
Boolean                 Boolean             Bool.h
Integer                 Integer             Integer.h
Real                    Real                Real.h
OctetString             OctetString         OctetString.h
GraphicString           GraphicString       GraphicString.h
IA5String               IA5String           IA5String.h
PrintableString         PrintableString     PrintableString.h
IntegerList             IntegerList         IntegerList.h
```

```
RealList                RealList                RealList.h
GraphicStringList       GraphicStringList       GraphicStringList.h
IA5StringList           IA5StringList           IA5StringList.h
PrintableStringList     PrintableStringList     PrintableStringList.h
Time                    Time                    Time.h
UTCTime                 Time                    Time.h
GeneralizedTime         Time                    Time.h
CmisScope               MScope                  Scope.h
CMISScope               MScope                  Scope.h
Scope                   MScope                  Scope.h
CmisFilter              MFilter                 Filter.h
CMISFilter              MFilter                 Filter.h
Filter                  MFilter                 Filter.h
ObjectIdentifier        ObjId                   ObjId.h
ObjectIdentifierList    ObjIdList               ObjIdList.h
ObjectClass             ObjectClass             ObjectClass.h
ObjectClassList         ObjectClassList         ObjectClassList.h
ObjectInstance          ObjectInstance          ObjectInstance.h
ObjectInstanceList      ObjectInstanceList      ObjectInstanceList.h
DistinguishedName       DName                   DName.h
EventTypeId             EventType               EventType.h
AttributeId             AttributeId             AttributeId.h
TimePeriod              TimePeriod              TimePeriod.h

#
# The following are syntaxes for attributes that may be re-used
#
AdministrativeState     AdministrativeState     AdministrativeState.h
OperationalState        OperationalState        OperationalState.h
UsageState              UsageState              UsageState.h
AvailabilityStatus      AvailabilityStatus      AvailabilityStatus.h

#
# The following are ATM-implemented syntaxes that may be re-used
#
BurstTolerance          BurstTolerance          BurstTolerance.h
CDVTolerance            CDVTolerance            CDVTolerance.h
OAMPeakCellRate         OAMPeakCellRate         OAMPeakCellRate.h
PeakCellRate            PeakCellRate            PeakCellRate.h
QosClass                QosClass                QosClass.h
SustainableCellRate     SustainableCellRate     SustainableCellRate.h

#
# The following are M.3100 implemented syntaxes that may be re-used
#
AlarmStatus                 AlarmStatus                 AlarmStatus.h
CurrentProblemList          CurrentProblemList          CurrentProblemList.h
SupportableClientList       ObjectClassList             ObjectClassList.h
SystemTimingSource          SystemTimingSource          SystemTimingSource.h
PointerOrNull               PointerOrNull               PointerOrNull.h
ConnectivityPointer         ConnectivityPointer         ConnectivityPointer.h
DownstreamConnectivityPointer DownstreamConnectivityPointer DownstreamConnectivityPointer.h
CrossConnectionObjectPointer CrossConnectionObjectPointer CrossConnectionObjectPointer.h
ObjectList                  ObjectList                  ObjectList.h
SystemTitle                 SystemTitle                 SystemTitle.h
MultipleConnections         MultipleConnections         MultipleConnections.h

##
Version                 GraphicString           GraphicString.h
UserLabel               GraphicString           GraphicString.h
VendorName              GraphicString           GraphicString.h
```

ExternalTime	Time		Time.h
ChannelNumber	Integer		Integer.h
CharacteristicInformation	ObjId	ObjId.h	
LocationName	GraphicString		GraphicString.h

11.10 Especificação GDMO do Elemento de Rede ATM

```
--#############################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: MIB.gdmo - AtmSource
                Descrição: Contém a especificação em GDMO das classes de objetos associados ao Elemento
                de Rede ATM.
--#############################################################################################

-- ############ Managed Object templates ###################

tcAdaptorTTPBidirectional MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T M.3100:1992":trailTerminationPointBidirectional;
        CHARACTERIZED BY

-- Conditional Packages
                -- alarmSeverityAssignmentPointerPackage,
                cellScramblingEnabledPkg,
-- ------------------------------------------------------------
                -- tmnCommunicationAlarmInformationPackage,
                -- createDeleteNotificationPackage,
                -- stateChangeNotificationPackage,
tcAdaptorTTPBidirectionalPkg PACKAGE
    BEHAVIOUR tcAdaptorTTPBidirectionalBeh;
    ATTRIBUTES
        tcTTPId     GET;;;
-- CONDITIONAL PACKAGES
-- "ITU-T M.3100:1992":alarmSeverityAssignmentPointerPackage
        -- PRESENT IF "the managed objct supports configuration of alarm severity",
-- cellScramblingEnabledPkg
        -- PRESENT IF "cell scrambling may be activated and deactivated for the supporting ATM interface.";
REGISTERED AS {atmManagedObjectClass 1};

------------------------------------------------------------------------------
---Alterado 14/02/97 - Ariadne

vcCTPBidirectional MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T M.3100:1992":
                connectionTerminationPointBidirectional;
        CHARACTERIZED BY

-- Conditional Packages
trafficDescriptorPkg,
oamTrafficDescriptorPkg,
-- "ITU-T M.3100:1992":AdministrativeStatePkg,
qosClassPkg,
oamCellLoopbackPkg,

-- ------------------------------------------------

        ITU-T M.3100:1992":attributeValueChangeNotificationPackage,
        "U-T M.3100:1992":createDeleteNotificationsPackage,
--      "ITU-T M.3100:1992":crossConnectionPointerPackage,
        vcCTPBidirectionalPkg PACKAGE
                BEHAVIOUR vcCTPBidirectionalBeh;
                ATTRIBUTES
```

```
                    vcCTPId                 GET,
                    segmentEndPoint DEFAULT VALUE AEmMod.booleanFalseDefault
                                    GET-REPLACE;;;

-- CONDITIONAL PACKAGES
-- trafficDescriptorPkg
-- PRESENT IF "This package must be present when the upstreamConnectivityPointer and
downstreamConnectivityPointer attributes point to an instance of the vcTTPBidirectional object class.",
-- oamTrafficDescriptorPkg
-- PRESENT IF "This package must be present when the   upstreamConnectivityPointer attributes point to an
instance of the vcTTPBidirectional object class.",
-- "ITU-T M.3100:1992":AdministrativeStatePkg
-- PRESENT IF "supported by the Network Element",
--qosClassPkg
-- PRESENT IF "QOS Class Information is supplied by the managing          system",
-- oamCellLoopbackPkg
-- PRESENT IF "the link termination point supports OAM cell Loopbacks";
REGISTERED AS {atmManagedObjectClass 2};

------------------------------------------------------------------------
vcTTPBidirectional MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T M.3100:1992":trailTerminationPointBidirectional;
        CHARACTERIZED BY

-- Conditional Packages

 oamCellLoopbackPkg,

-- ------------------------------------------------
-- "Rec. X.721|ISO/IEC 10165-2":administrativeStatePackage,
-- "ITU-T M.3100:1992":attributeValueChangeNotificationPackage,
-- "ITU-T M.3100:1992":createDeleteNotificationPackage,

vcTTPBidirectionalPkg PACKAGE
    BEHAVIOUR vcTTPBidirectionalBeh;
    ATTRIBUTES
        vcTTPId     GET;;;
-- CONDITIONAL PACKAGES
-- oamCellLoopbackPkg
    -- PRESENT IF "the VCC termination point supports OAM cell Loopbacks";
REGISTERED AS {atmManagedObjectClass 3};

------------------------------------------------------------------------
vpCTPBidirectional MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T M.3100:1992":
                connectionTerminationPointBidirectional;
        CHARACTERIZED BY

-- Conditional Packages
trafficDescriptorPkg,
oamTrafficDescriptorPkg,
-- "ITU-T M.3100:1992":AdministrativeStatePkg,
qosClassPkg,
oamCellLoopbackPkg,
-- ------------------------------------------------

--      "ITU-T M.3100:1992":attributeValueChangeNotificationPackage,
--      "ITU-T M.3100:1992":createDeleteNotificationsPackage,
--      "ITU-T M.3100:1992":crossConnectionPointerPackage,
        vpCTPBidirectionalPkg PACKAGE
```

```
                BEHAVIOUR vpCTPBidirectionalBeh;
                ATTRIBUTES
                        vpCTPId                 GET,
                        segmentEndPointDEFAULT VALUE
                                                AtmMIBMod.booleanFalseDefault
                                        GET-REPLACE;;;


        -- CONDITIONAL PACKAGES
-- trafficDescriptorPkg
-- PRESENT IF "supplied by the managing system. This package must be present at points where UPC/NPC
functions are performed or when the upstreamConnectivityPointer and downstreamConnectivity Pointer attributes
point to an instance of the vpTTPBidirectional object class",
-- oamTrafficDescriptorPkg
-- PRESENT IF "supplied by the managing system. This package must be present at points where UPC/NPC
functions are performed or when the upstreamConnectivityPointer and downstreamConnectivity Pointer attributes
point to an instance of the vpTTPBidirectional object class",
-- "ITU-T M.3100:1992":AdministrativeStatePkg
-- PRESENT IF "supported by the Network Element",
-- qoClassPkg
-- PRESENT IF "QOS Class Information is supplied by the managing system",
-- oamCellLoopbackPkg
-- PRESENT IF "the VPL termination point supports OAM cell Loopbacks";

REGISTERED AS {atmManagedObjectClass 4};

------------------------------------------------------------------------------
vpTTPBidirectional MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T M.3100:1992":trailTerminationPointBidirectional;
        CHARACTERIZED BY
-- Conditional Packages
oamCellLoopbackPkg,
-- ------------------------------------------------

-- "Rec. X.721|ISO /IEC 10165-2":administrativeStatePackage,
-- "ITU-T M.3100:1992":attributeValueChangeNotificationPackage,
-- "ITU-T M.3100:1992":createDeleteNotificationPackage,
vpTTPBidirectionalPkg PACKAGE
        BEHAVIOUR vpTTPBidirectionalBeh;
        ATTRIBUTES
                vpTTPId                 GET;;;
-- CONDITIONAL PACKAGES
-- oamCellLoopbackPkg
        -- PRESENT IF "the VPC termination point supports OAM cell Loopbacks";
REGISTERED AS {atmManagedObjectClass 5};

------------------------------------------------------------------------------



-- ############ Packages templates ##########################

attributeValueChangeNotificationPackage   PACKAGE
        NOTIFICATIONS
                attributeValueChange;
REGISTERED AS {m3100Package 4};

createDeleteNotificationsPackage PACKAGE
        NOTIFICATIONS
                objectCreation,
                objectDeletion;
```

```
REGISTERED AS {m3100Package 10};


cellScramblingEnabledPkg PACKAGE
        ATTRIBUTES
                cellScramblingEnabled
                        GET-REPLACE;
REGISTERED AS {atmPackage 1};


oamCellLoopbackPkg PACKAGE
        ACTIONS
                loopbackOAMCell;
REGISTERED AS {atmPackage 2};


oamTrafficDescriptorPkg PACKAGE
        ATTRIBUTES
                oamIngressPeakCellRate
                        GET-REPLACE,
                oamEgressPeakCellRate
                        GET-REPLACE,
                oamIngressCDVTolerance
                        GET-REPLACE,
                oamEgressCDVTolerance
                        GET-REPLACE;
REGISTERED AS {atmPackage 3};


qosClassPkg PACKAGE
        ATTRIBUTES
                ingressQOSClass
                        GET,
                egressQOSClass
                        GET;
REGISTERED AS {atmPackage 4};


trafficDescriptorPkg PACKAGE
        ATTRIBUTES
                ingressPeakCellRate
                        GET-REPLACE
                        ADD-REMOVE,
                egressPeakCellRate
                        GET-REPLACE
                        ADD-REMOVE,
                ingressCDVTolerance
                        GET-REPLACE
                        ADD-REMOVE,
                egressCDVTolerance
                        GET-REPLACE
                        ADD-REMOVE,
                ingressSustainableCellRate
                        GET-REPLACE
                        ADD-REMOVE,
                egressSustainableCellRate
                        GET-REPLACE
                        ADD-REMOVE,
                ingressBurstTolerance
                        GET-REPLACE
                        ADD-REMOVE,
```

```
                egressBurstTolerance
                        GET-REPLACE
                        ADD-REMOVE;
REGISTERED AS {atmPackage 5};



-- ############ Attributes templates ##########################

cellScramblingEnabled ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MetricModule.Boolean;
        MATCHES FOR EQUALITY;
        BEHAVIOUR cellScramblingEnabledBeh;
REGISTERED AS {atmAttributeID 1};

egressBurstTolerance ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.BurstTolerance;
        MATCHES FOR EQUALITY;
        BEHAVIOUR egressBurstToleranceBeh;
REGISTERED AS {atmAttributeID 2};

egressCDVTolerance ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.CDVTolerance;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR egressCDVToleranceBeh;
REGISTERED AS {atmAttributeID 3};

egressPeakCellRate ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.PeakCellRate;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR egressPeakCellRateBeh;
REGISTERED AS {atmAttributeID 4};

egressQOSClass ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.QosClass;
        MATCHES FOR EQUALITY;
        BEHAVIOUR egressQOSClassBeh;
REGISTERED AS {atmAttributeID 5};

egressSustainableCellRate ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.SustainableCellRate;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR egressSustainableCellRateBeh;
REGISTERED AS {atmAttributeID 6};

ingressBurstTolerance ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.BurstTolerance;
        MATCHES FOR EQUALITY;
        BEHAVIOUR ingressBurstToleranceBeh;
REGISTERED AS {atmAttributeID 7};

ingressCDVTolerance ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.CDVTolerance;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR ingressCDVToleranceBeh;
REGISTERED AS {atmAttributeID 8};

ingressPeakCellRate ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.PeakCellRate;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR ingressPeakCellRateBeh;
```

```
REGISTERED AS {atmAttributeID 9};

ingressQOSClass ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.QosClass;
        MATCHES FOR EQUALITY;
        BEHAVIOUR ingressQOSClassBeh;
REGISTERED AS {atmAttributeID 10};

ingressSustainableCellRate ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.SustainableCellRate;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR ingressSustainableCellRateBeh;
REGISTERED AS {atmAttributeID 11};

oamEgressCDVTolerance ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1TypeModule.OAMCDVTolerance;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR oamEgressCDVToleranceBeh;
REGISTERED AS {atmAttributeID 14};

oamEgressPeakCellRate ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1TypeModule.OAMPeakCellRate;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR oamEgressPeakCellRateBeh;
REGISTERED AS {atmAttributeID 15};

oamIngressCDVTolerance ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1TypeModule.OAMCDVTolerance;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR oamIngressCDVToleranceBeh;
REGISTERED AS {atmAttributeID 12};

oamIngressPeakCellRate ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASN1TypeModule.OAMPeakCellRate;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR oamIngressPeakCellRateBeh;
REGISTERED AS {atmAttributeID 13};

segmentEndPoint ATTRIBUTE
        WITH ATTRIBUTE SYNTAX AtmMIBMod.Boolean;
        MATCHES FOR EQUALITY;
        BEHAVIOUR segmentEndPointBeh;
REGISTERED AS {atmAttributeID 16};

tcTTPId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.SimpleNameType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR tcTTPIdBeh;
REGISTERED AS {atmAttributeID 17};

vcCTPId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.SimpleNameType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR vcCTPIdBeh;
REGISTERED AS {atmAttributeID 18};

vcTTPId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.SimpleNameType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR vcTTPIdBeh;
REGISTERED AS {atmAttributeID 19};
```

```
vpCTPId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.SimpleNameType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR vpCTPIdBeh;
REGISTERED AS {atmAttributeID 20};

vpTTPId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX UCLAttribute-ASN1Module.SimpleNameType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR vpTTPIdBeh;
REGISTERED AS {atmAttributeID 21};


-- ############ Notifications and actions templates ##########################

attributeValueChange NOTIFICATION
  BEHAVIOUR              attributeValueChangeBehaviour;
  WITH INFORMATION SYNTAX   Notification-ASN1Module.AttrValChangeInfo
  AND ATTRIBUTE IDS
    sourceIndicator      sourceIndicator,
    attrValChangeDefinition attrValChangeDefinition;
  REGISTERED AS          { smi2Notification 1 }

loopbackOAMCell ACTION
        BEHAVIOUR loopbackOAMCellBeh;
        MODE CONFIRMED;
        WITH INFORMATION SYNTAX AtmMIBMod.LoopbackOAMCellInfo;
        WITH REPLY SYNTAX AtmMIBMod.LoopbackOAMCellReply;
REGISTERED AS {atmAction 1};

objectCreation NOTIFICATION
  BEHAVIOUR              objectCreationBehaviour;
  WITH INFORMATION SYNTAX   Notification-ASN1Module.ObjectInfo
  AND ATTRIBUTE IDS
    sourceIndicator      sourceIndicator,
    attributeList        attributeList;
  REGISTERED AS          { smi2Notification 6 };

objectDeletion NOTIFICATION
  BEHAVIOUR              objectDeletionBehaviour;
  WITH INFORMATION SYNTAX   Notification-ASN1Module.ObjectInfo
  AND ATTRIBUTE IDS
    sourceIndicator      sourceIndicator,
    attributeList        attributeList;
  REGISTERED AS          { smi2Notification 7 };


-- ############## Name Binding Templates ##############


tcAdaptorTTPBidirectional-managedElementR1 NAME BINDING
  SUBORDINATE OBJECT CLASS   tcAdaptorTTPBidirectional AND SUBCLASSES;
  NAMED BY
  SUPERIOR OBJECT CLASS     "ITU-T M.3100:1992":managedElementR1 AND SUBCLASSES;
  WITH ATTRIBUTE          tcTTPId;
  REGISTERED AS          {atmNameBindings 1};

---
vcCTPBidirectional-vpTTPBidirectional NAME BINDING
  SUBORDINATE OBJECT CLASS   vcCTPBidirectional AND SUBCLASSES;
```

```
  NAMED BY
  SUPERIOR OBJECT CLASS   vpTTPBidirectional AND SUBCLASSES;
  WITH ATTRIBUTE          vcCTPId;
  CREATE                  WITH-AUTOMATIC-INSTANCE-NAMING;
  DELETE                  ONLY-IF-NO-CONTAINED-OBJECTS;
  REGISTERED AS           {atmNameBindings 2};

---
vcTTPBidirectional-managedElementR1    NAME BINDING
  SUBORDINATE OBJECT CLASS        vcTTPBidirectional AND SUBCLASSES;
  NAMED BY
  SUPERIOR OBJECT CLASS   "ITU-T M.3100:1992":managedElementR1 AND SUBCLASSES;
  WITH ATTRIBUTE          vcTTPId;
  CREATE                  WITH-AUTOMATIC-INSTANCE-NAMING;
  DELETE                  ONLY-IF-NO-CONTAINED-OBJECTS;
  REGISTERED AS           {atmNameBindings 3};


---
vpCTPBidirectional-tcAdaptorTTPBidirectional NAME BINDING
  SUBORDINATE OBJECT CLASS        vpCTPBidirectional AND SUBCLASSES;
  NAMED BY
  SUPERIOR OBJECT CLASS   tcAdaptorTTPBidirectional AND SUBCLASSES;
  WITH ATTRIBUTE          vpCTPId;
  CREATE                  WITH-AUTOMATIC-INSTANCE-NAMING;
  DELETE                  DELETES-CONTAINED-OBJECTS;
  REGISTERED AS           {atmNameBindings 4};

---
vpTTPBidirectional-managedElementR1 NAME BINDING
  SUBORDINATE OBJECT CLASS   vpTTPBidirectional AND SUBCLASSES;
  NAMED BY
  SUPERIOR OBJECT CLASS     "ITU-T M.3100:1992":managedElementR1 AND SUBCLASSES;
  WITH ATTRIBUTE          vpTTPId;
  CREATE                  WITH-AUTOMATIC-INSTANCE-NAMING;
  DELETE                  ONLY-IF-NO-CONTAINED-OBJECTS;
  REGISTERED AS           {atmNameBindings 5};

---

-- ############ Behaviours templates ##########################

cellScramblingEnabledBeh BEHAVIOUR
DEFINED AS "This attribute identifies whether or not ATM cell scrambling is being performed over the ATM
interface. Avalue of TRUE (default) is used to indicate that cell scrambling is being performed.";

---
egressBurstToleranceBeh BEHAVIOUR
DEFINED AS "This attribute represents the egress (with respect to the managed
system) burst tolerance (in cells) that has been assigned to the VP or VC link
being terminated.";

---
egressCDVToleranceBeh BEHAVIOUR
DEFINED AS "This attribute represents the egress (with respect to the managed
system) CDV Tolerance assigned to the VPL or VCL being terminated. The default
value for this attribute is 0.";

---
egressPeakCellRateBeh   BEHAVIOUR
```

DEFINED AS "This attribute is used to indicate the peakCellRate assigned or reserved in the egress (with respect to the managed system) direction of
transmission across the VP or VC link being terminated.";

---
egressQOSClassBeh BEHAVIOUR
DEFINED AS "This attribute identifies the Quality Of Service (QOS) class assigned to the VPL or VCL in the Egress (with respect to the managed system) direction of cell transmission. Valid values for this attribute are: Class 1, Class 2, Class 3 and Class 4.";

---
egressSustainableCellRateBeh BEHAVIOUR
DEFINED AS "This traffic descriptor represents the egress (with respect to the managed system) sustainable cell rate (in cells/second) assigned to the link being terminated.";

---
ingressBurstToleranceBehBEHAVIOUR
DEFINED AS "This attribute represents the ingress (with respect to the managed system) burst tolerance (in cells) that has been assigned to the VP or VC link being terminated.";

---
ingressCDVToleranceBeh BEHAVIOUR
DEFINED AS "This attribute represents the ingress (with respect to the managed system) CDV Tolerance assigned to the VPL or VCL being terminated. The default value for this attribute is 0.";

---
ingressPeakCellRateBeh BEHAVIOUR
DEFINED AS "This attribute is used to indicate the peakCellRate assigned or reserved in the ingress (with respect to the managed system) direction of
transmission across the VP or VC link being terminated.";

---
ingressQOSClassBeh BEHAVIOUR
DEFINED AS "This attribute identifies the Quality Of Service (QOS) class assigned to the VPL or VCL in the ingress (with respect to the managed system) direction of cell transmission. Valid values for this attribute are: Class 1, Class 2, Class 3 and Class 4.";

---
ingressSustainableCellRateBeh BEHAVIOUR
DEFINED AS "This traffic descriptor represents the ingress (with respect to the managed system) sustainable cell rate (in cells/second) assigned to the link being terminated.";

---
loopbackOAMCellBeh BEHAVIOUR
DEFINED AS "This action is used to request a vpCTPBidirectional, vcCTPBidirectional, vpTTPBidirectional, or vcTTPBidirectional object to insert (in the outgoing direction) a loopback OAM cell into the ATM cell stream and verify its return.

Supplied along with this action is the loopbackLocation parameter. This parameter identifies the downstream vpCTPBidirectional, vcCTPBidirectional, vpTTPBidirectional, or vcTTPBidirectional object instance responsible for looping back the OAM cell. The value of TRUE-NULL (default) can be used to request the end-point of the ATM connection or connection segment to loopback the OAM cell. Also supplied with this parameter is an indication as to whether or not the OAM Loopback Cell to be inserted shall be of the segment type of end-to-end type.";

---
oamEgressCDVToleranceBeh BEHAVIOUR

DEFINED AS "This attribute is used to indicate the oam cell delay variation assigned or reserved in the oam egress direction.";

---

oamEgressPeakCellRateBeh BEHAVIOUR
DEFINED AS "This attribute is used to indicate the oam peak cell rate assigned or reserved in the oam egress direction.";

---

oamIngressCDVToleranceBeh BEHAVIOUR
DEFINED AS "This attribute is used to indicate the oam cell delay variation assigned or reserved in the oam ingress direction.";

---

oamIngressPeakCellRateBeh BEHAVIOUR
DEFINED AS "This attribute is used to indicate the oam peak cell rate assigned or reserved in the oam ingress direction.";

---

segmentEndPointBeh BEHAVIOUR
DEFINED AS "This attribute contains the average monitoring block size, associated with the outgoing direction of a monitored ATM cell flow along a
VPC/VCC. The outAverageMonitoringBlockSize may be set to a lenght of 128, 256, 512 or 1024 cells. This attribute is automatically set as result of the VPC performance monitoring activation.";

---

tcAdaptorTTPBidirectionalBeh BEHAVIOUR
DEFINED AS "This managed object represents a point in the managed system where the datation ofthe ATM layer to the underlying physical infrastructure (e.g., SDH or PDH transport network) take place. ITU-T Recommendation I.321[12] identifies this adaptation function as one of many functions performed at the Transmission Convergence (TC) Sublayer of the BISDN protocol stack.

This object is responsible for generating communicationsAlarm notifications that report the inability of the managed system to delineate ATM cellfrom the payload of a terminated digital transmission path.

The supportedByObjectList attribute inherited the trailTerminationPoint managed object shall include a pointer to the underlying, path-level trail terminatio point managed (e.g., vcTTPBidirectional object).";

---

tcTTPIdBeh BEHAVIOUR
DEFINED AS "This attribute is used for naming instances of the tcAdaptorTTPBidirectional managed object class.";

---

vcCTPBidirectionalBeh BEHAVIOUR
DEFINED AS "The vcCTPBidirectional object class is a class of managed objects that delimit Virtual Channel (VC) links. From a configuration management perspective, instances of this object class represent VC link terminations that are either cross-connected to other VC link terminations or are avaible for such cross-connection.

Instances of this object class include attributes that describe
the VCI value, traffic descriptors, and, optionally, the Quality of Service (QOS) class assigned to the VCL termination being represented. Note that the vcCTPId attribute value identifies the VCI value for the VCL being terminated and is also used as the RDN for naming instances of this object class. The vcCTPId attribute value may be provided by the managing system upon creation of this managed object instance or it may be absent in M-CREATE message and thus selected by the managed system.

When selected by the managed system, the value chosen shall be reported to the managing system as a parameter in the response to the successfully carried out M-CREATE request.

From a performance and fault management perspective, instances of this object class represent logical points along VCCs at which various maintenance and network traffic management functions may be performed.

In the event that the related vcTTPBidirectional is created, this instance points to the vcTTPBidirectional.

The conditional package oamCellLoopbackPkg provides the M-ACTION used to request the termination point to insert an OAM cell for downstream loopbacking and report whether or not cell was returned within the required time.

The administrativeState attribute may be used by the management system to inhibit (lock) and allow (unlock) the flow of cells through the vcCTPBidirectional. However, segment OAM cell shall be passed to/from the contained monitor object (if any), so that vcCTPBidirectional can be used as a segment end-point.

Instances of this object class may be explicitly created and deleted by the managing system using the CMIS M-CREATE and M-DELETE services, respectively.";

---

vcTTPBidirectionalBeh BEHAVIOUR
DEFINED AS "The vcTTPBidirectional object is a class of managed that delimit Virtual Channel Connections (VCCs).

An instance of this object represent the logical point in the managed system where the end-to-end F5 flow (i.e., OAM cells with PT=5) terminates.

The condictional package oamCellLoopbackPKg provides the M-ACTION used to request the termination point to insert OAM cell for dowstream loopbacking and report whether or not the cell was returned within the required time.

Instances of this object class are explicity created and deleted by the managing system using the CMIS M-CREATE and M-DELETE services, respectively.

An instance os this object shall always point to vcCTPCBideirectional managed objetct using UCP/UDP.";

---

vcCTPIdBeh BEHAVIOUR
DEFINED AS "This attribute is used for naming instances of the vcCTPBidirectional managed object class. The value of this attribute shall be set equal to the VCI value of the Virtual Channel Link (VCL) being terminated.";

---

vcTTPIdBeh BEHAVIOUR
DEFINED AS "This attribute is used for naming instances of the vcTTPBidirectional managed object class.";

---

vpCTPBidirectionalBeh BEHAVIOUR
DEFINED AS "The vpCTPBidirectional object class of managed objects that delimit Virtual Path (VP) links. From a configuration management perspective, instances of this object class represent VP link terminations that are either cross-connected to other VP link terminations or are available for such cross-connection.

Instances of this object class include attributes that describe the VPI value, traffic descriptors, and, optionally, the Quality of Service (QOS) class assigned to the VPL termination being represented. Note that the vpCTPId attribute value identifies the VPI value of the VPL termination being represented and is also used as the RDN for naming instances of this object class.

Note that vpCTPId attribute may be provided by the managing system upon creation of this managed object instance or it may be absent in M-CREATE message and thus selected by the managed system. When selected by the managed system, the value chosen shall be reported to the managing system as a parameter in the response to the successfully carried out M-CREATE request.

From a performance and fault management perspective, instances of this object class represent logical points along VPCs at which various maintenance and network traffic management functions may be performed.

In the event that the related vpTTPBidirectional is created, this instance points to the vcTTPBidirectional and its crossConnectionPointer points to the atmFabric instance.

The conditional package oamCellLoopbackPkg provides the M-ACTION used to request the termination point to insert an OAM cell for downstream loopbacking and report whether or not cell was returned within the required time.

The administrativeState attribute may be used by the management system to inhibit (lock) and allow (unlock) the flow of cells through the vcCTPBidirectional. However, segment OAM cell shall be passed to/from the contained monitor object (if any), so that vpCTPBidirectional can be used as a segment end-point.

When a VP-AIS or VP-RDI failure is detected, the vpCTPBidirectional object shall generate a communicationsAlarm notification(if the tmnCommunicationsAlarmInformationPackage is present) with the probableCause parameter value set equal to aIS or farEndReceiveFailure, respectively.

Instances of this object class may explicitly created and deleted by the managing system using the CMIS M-CREATE and M-DELETE services, respectively. Instances of this managed object class may also be automatically created by the managed system in r

11.11 Especificação ASN.1 das Sintaxes de Atributos Associadas ao Elemento de Rede ATM

```
--#############################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: AtmMIBMod.py - AtmSource
                Descrição: Contém a especificação em ASN1 das sintaxes de atributos associados ao Elemento
                de Rede ATM.
--#############################################################################

AtmMIBMod -- { }

DEFINITIONS IMPLICIT TAGS ::=

%{
#include <stdio.h>
#ifndef  ICR1
#include <isode/isoaddrs.h>
#include <isode/pepsy/ACS-types.h>
#else
#include <isode/ll/isoaddrs.h>
#include <isode/asn1/ACS-types.h>
#endif
%}


PREFIXES build parse print

BEGIN

IMPORTS
        PSAPaddr FROM DSE
        AE-title FROM ACS
        --DistinguishedName, RDNSequence FROM IF
        --AttributeId, AttributeList, ObjectClass, ObjectInstance, Scope FROM CMIP
        ProblemCause FROM ASN1DefinedTypesModule
        GraphicString, SimpleNameType FROM SMI {
                joint-iso-ccitt
                ms(9)
                part2(2)
                asn1Module(2)
                1
        };


SECTIONS build parse print

--supporting productions

BurstTolerance ::= SEQUENCE {
        burstToleranceCLP0plus1 [1] INTEGER OPTIONAL,
        burstToleranceCLP0 [2] INTEGER OPTIONAL}

CDVTolerance ::= SEQUENCE {
        cellDelayVariationToleranceCLP0plus1 [1] INTEGER OPTIONAL,
        cellDelayVariationToleranceCLP0 [2] INTEGER OPTIONAL}
```

```
GraphicStringOrNull ::= CHOICE {
        graphicString GraphicString,
        null NULL}

LoopbackLocation ::= SEQUENCE {
        endPoint BOOLEAN, --default is TRUE
        locationId GraphicStringOrNull} --default is NULL

LoopbackOAMCellInfo ::= SEQUENCE {
        loopbackLocation LoopbackLocation,
        oamCellType OamCellType}

OamCellType ::= ENUMERATED {
        segment (0),
        endToEnd (1)}

PeakCellRate ::= SEQUENCE {
        peakCellRateCLPplus1 [1] INTEGER OPTIONAL,
        peakCellRateCLP0 [2] INTEGER OPTIONAL}


QosClass ::= ENUMERATED {
        class1 (0),
        class2 (1),
        class3 (2),
        class4 (3)}

--Class A Level
--Class B Level
--Class C Level
--Class D Level



SustainableCellRate ::= SEQUENCE {
        sustainableCellRateCLP0plus1 [1] INTEGER OPTIONAL,
        sustainableCellRateCLP0 [2] INTEGER OPTIONAL}

LoopbackOAMCellReply ::= SEQUENCE {
        loopbackSuccessfull BOOLEAN,
        problemCause ProblemCause OPTIONAL}


END
```

11.12 Classes de Atributos Associadas ao Elemento de Rede ATM

```
--#############################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: Attr.dat - AtmSource
                Descrição: Contém as classes de atributos associadas ao Elemento de Rede ATM.
--#############################################################################################
#
#     attr.dat is a database of attribute classes
#     The fields of each record (line) in the file are:
#
# 1) GDMO Attribute Type Name 2) Attribute C++ Class Name 3) C++ Header file
#
#
#############################################################
#
# These are commonly used attributes from the DMI document
# (ISO 10165-2 / CCITT X.721) which may be referred to from other GDMO specs
#
objectClass                 ObjectClass             ObjectClass.h
managedObjectClass          ObjectClass             ObjectClass.h
managedObjectInstance       ObjectInstance          ObjectInstance.h
eventType                   EventType               EventType.h
discriminatorConstruct      MFilter                 Filter.h
destination                 DestinationAddressGms   Destination.h
administrativeState         AdministrativeState     AdministrativeState.h
operationalState            OperationalState        OperationalState.h
usageState                  UsageState              UsageState.h
availabilityStatus          AvailabilityStatus      AvailabilityStatus.h
systemTitle                 SystemTitle             SystemTitle.h

#
# These are the generic attribute types from which other may be derived
# using the DERIVED FROM GDMO clause
#
cellScramblingEnabled       Boolean                 Bool.h
egressBurstTolerance        BurstTolerance          BurstTolerance.h
egressCDVTolerance          CDVTolerance            CDVTolerance.h
egressPeakCellRate          PeakCellRate            PeakCellRate.h
egressQOSClass              QosClass                QosClass.h
egressSustainableCellRate   SustainableCellRate     SustainableCellRate.h
ingressBurstTolerance       BurstTolerance          BurstTolerance.h
ingressCDVTolerance         CDVTolerance            CDVTolerance.h
ingressPeakCellRate         PeakCellRate            PeakCellRate.h
ingressQOSClass             QosClass                QosClass.h
ingressSustainableCellRate  SustainableCellRate     SustainableCellRate.h
oamEgressCDVTolerance       CDVTolerance            CDVTolerance.h
oamEgressPeakCellRate       PeakCellRate            PeakCellRate.h
oamIngressCDVTolerance      CDVTolerance            CDVTolerance.h
oamIngressPeakCellRate      PeakCellRate            PeakCellRate.h
segmentEndPoint             Boolean                 Bool.h
tcTTPId                     SimpleNameType          SimpleNameType.h
vcCTPId                     SimpleNameType          SimpleNameType.h
vcTTPId                     SimpleNameType          SimpleNameType.h
vpCTPId                     SimpleNameType          SimpleNameType.h
vpTTPId                     SimpleNameType          SimpleNameType.h
```

11.13 Sintaxes dos Atributos Associadas ao Elemento de Rede ATM

```
--#############################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: Syntax.dat - AtmSource
                Descrição: Contém as sintaxes dos atributos associados ao Elemento de Rede ATM.
--#############################################################################################
#
#     syntax.dat is a database of syntax classes
#     The fields of each record (line) in the file are:
#
# 1) GDMO Syntax Name 2) C++ Class Name 3) C++ Header file
#
#
# INTEGER                       Integer             Integer.h
# REAL                          Real                Real.h
# OCTET STRING                  OctetString         OctetString.h
# IA5String                     String              IA5String.h
# SET OF GraphicString          StringList          StringList.h
# SET OF IA5String              IA5StringList       IA5StringList.h
# UTCTime                       Time                Time.h
# CMISFilter                    MFilter             Filter.h
# Scope                         MScope              Scope.h
# OBJECT IDENTIFIER             ObjId               ObjId.h
# SET OF OBJECT IDENTIFIER      ObjIdList           bjIdList.h
# ObjectClass                   ObjectClass         ObjectClass.h
# SET OF ObjectClass            ObjectClassList     ObjectClassList.h
#
############################################################
#
# The following should not be used if the DERIVED FROM clause is used
# for the generic attribute types [ counter, gauge, counter-Threshold
# gauge-Threshold and tide-Mark ] instead of the WITH ATTRIBUTE SYNTAX.
# In this case, the attr.dat file contains the right entries.
# In case though that this is not the case, the syntax of those
# attributes is included below.
#
Count                   Counter             Counter.h
CounterThreshold        CounterThreshold    CounterThld.h
ObservedValue           Gauge               Gauge.h
GaugeThreshold          GaugeThreshold      GaugeTnld.h
TideMarkInfo            TideMark            TdeMark.h
#
# The following are OSIMIS-implemented syntaxes that may be re-used
#
SimpleNameType          SimpleNameType      SimpleNameType.h
Boolean                 Boolean             Bool.h
Integer                 Integer             Integer.h
Real                    Real                Real.h
OctetString             OctetString         OctetString.h
GraphicString           GraphicString       GraphicString.h
IA5String               IA5String           IA5String.h
PrintableString         PrintableString     PrintableString.h
IntegerList             IntegerList         IntegerList.h
RealList                RealList            RealList.h
GraphicStringList       GraphicStringList   GraphicStringList.h
```

IA5StringList	IA5StringList	IA5StringList.h
PrintableStringList	PrintableStringList	PrintableStringList.h
Time	Time	Time.h
UTCTime	Time	Time.h
GeneralizedTime	Time	Time.h
CmisScope	MScope	Scope.h
CMISScope	MScope	Scope.h
Scope	MScope	Scope.h
CmisFilter	MFilter	Filter.h
CMISFilter	MFilter	Filter.h
Filter	MFilter	Filter.h
ObjectIdentifier	ObjId	ObjId.h
ObjectIdentifierList	ObjIdList	ObjIdList.h
ObjectClass	ObjectClass	ObjectClass.h
ObjectClassList	ObjectClassList	ObjectClassList.h
ObjectInstance	ObjectInstance	ObjectInstance.h
ObjectInstanceList	ObjectInstanceList	ObjectInstanceList.h
DistinguishedName	DName	DName.h
EventTypeId	EventType	EventType.h
AttributeId	AttributeId	AttributeId.h
TimePeriod	TimePeriod	TimePeriod.h

#
The following are syntaxes for attributes that may be re-used
#

AdministrativeState	AdministrativeState	AdministrativeState.h
OperationalState	OperationalState	OperationalState.h
UsageState	UsageState	UsageState.h
AvailabilityStatus	AvailabilityStatus	AvailabilityStatus.h

#
The following are ATM-implemented syntaxes that may be re-used
#

BurstTolerance	BurstTolerance	BurstTolerance.h
CDVTolerance	CDVTolerance	CDVTolerance.h
OAMPeakCellRate	OAMPeakCellRate	OAMPeakCellRate.h
PeakCellRate	PeakCellRate	PeakCellRate.h
QosClass	QosClass	QosClass.h
SustainableCellRate	SustainableCellRate	SustainableCellRate.h
##		
OAMCDVTolerance	Integer	Integer.h

11.14 Especificação ASN.1 das Sintaxes de Atributos Associadas ao Elemento de Rede ATM

```
--#############################################################################################
                Universidade Federal de Santa Catarina
                Curso de Pós Graduação em Ciência da Computação

                Arquivo: ASN1TypeModule.py- AtmSource
                Descrição: Contém a especificação em ASN1 das sintaxes de atributos associadas ao Elemento
                de Rede ATM.
--#############################################################################################

ASN1TypeModule {ccitt(0) identified-organization(4) etsi(0) informationModel(0) asn1Module(2)
asn1TypesModule(0)}
DEFINITIONS IMPLICIT TAGS ::=

PREFIXES build parse print

BEGIN

SECTIONS build parse print

--CDVTolerance ::= INTEGER
OAMPeakCellRate ::= SEQUENCE {
                m INTEGER (0..31),
                k INTEGER (0..511)}

END
```