UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

Departamento de Informática

Suporte para vistas em ambientes de gestão de redes

Carlos David Soares da Palma

(Licenciado)

Dissertação para obtenção do grau de

Mestre em Informática

Setembro de 2000

Suporte para vistas em ambientes de gestão de redes

Carlos David Soares da Palma

Dissertação submetida para provas

de mestrado em

Informática

Departamento de Informática

Faculdade de Ciências da Universidade de Lisboa

Lisboa

Setembro de 2000

Dissertação realizada sob a orientação do

Doutor Luís Eduardo Teixeira Rodrigues

Professor Auxiliar

Faculdade de Ciências da Universidade de Lisboa

Resumo

Em sistemas de gestão de redes, uma *vista* representa uma perspectiva parcelar do universo de gestão, adequada a uma actividade ou cargo específico. O suporte para vistas assume uma especial relevância em sistemas de grande dimensão onde não é possível concentrar toda a actividade de gestão sobre um único técnico.

Esta dissertação aborda o problema de fornecer os mecanismos adequados para o desenvolvimento de plataformas de gestão de redes capazes de suportar simultaneamente diversas vistas dos mesmos objectos de gestão.

A partir de uma panorâmica do estado da arte nos sistemas de gestão de rede justifica-se a necessidade e a importância de um sistema de gestão de vistas e sintetizam-se os trabalhos feitos no sentido de suportar esta funcionalidade. Partindo da identificação de algumas lacunas nos sistemas anteriormente descritos, é feita uma proposta de um modelo de suporte á gestão de vistas. Este modelo tem em conta aspectos funcionais e operacionais, salientando-se algumas funcionalidades como a capacidade para suportar restrições ao comportamento sensíveis à vista e a capacidade de propagar eventos por e entre as diversas vistas.

De modo a avaliar a exequibilidade do modelo proposto, foi realizado um protótipo baseado na plataforma de gestão Tkined/Scotty. Este protótipo foi avaliado através da sua aplicação a uma rede alvo e da comparação do sistema de gestão resultante com a aplicação de outras plataformas de gestão ao mesmo caso de estudo.

Abstract

In network management systems a *view* represents a subset of the universe of managed components which is presented in a way that is meaningful for a specific management task. The support for *views* assumes a particular relevance in large-scale networks where it is not possible to concentrate all management activity in a single network manager.

The thesis studies the problem of providing the adequate mechanisms for the development of network management platforms that are able to support, simultaneously, different views of the same universe of managed objects.

From a survey describing the state-of-the art in network management platforms, the need and relevance of view management systems is motivated. The survey also summarizes previous attempts in the area of view support. A number of weakness of current systems are identified and the thesis proposes a new model for the support of views. The proposed model takes both functional and operational aspects into account, such as the ability to express management constrains that are view dependent and the ability to support the propagation of events within and across views.

To evaluate the proposed model, a prototype has been implemented as an extension to the Tkined/Scotty system. The prototype has been evaluated using a target network and comparing the resulting system with the result of applying other network management tools to the same target.

Palavras Chave

Gestão de Redes

Gestão distribuída

Suporte para vistas

Keywords

Network Management

Distributed management

View support

Agradecimentos

As minhas primeiras palavras de agradecimento vão para o Professor Luís Rodrigues, pelos conselhos e orientações que me foi dando ao longo deste trabalho, pela sua capacidade em olhar para os problemas de uma forma positiva, pelo seu interesse genuíno pelo meu trabalho.

Também agradeço ao Dr. Hugo Miranda pelos comentários e sugestões.

Por fim agradeço à minha mulher São e à minha filha Ana Sofia, a compreensão, o apoio e a paciência que mais uma vez demonstraram.

Lisboa, Setembro de 2000.

Carlos David Soares da Palma

(Este trabalho é dedicado à São e à Ana Sofia).

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Uma rede informática complexa possui um elevado número de componentes (computadores, dispositivos periféricos, aplicações, etc), cada qual caracterizado por um conjunto variado de atributos que são acedidos e manipulados nas actividades de gestão da rede. Apesar da informação que caracteriza uma rede complexa poder atingir dimensões consideráveis, raramente as actividades de gestão da rede informática, necessitam de toda a informação disponivel. Pelo contrário, torna-se útil definir pontos de vista, ou simplesmente vistas, que seleccionam os componentes (e os atributos) relevantes para cada actividade específica de gestão da rede ou do sistema de informação.

Por exemplo, considere-se uma rede que possui máquinas executando o sistema operativo Unix e outras executando Windows NT, distribuídas fisicamente por diversas salas e ligadas a diferentes segmentos de rede IP. Para certas actividades, é interessante organizar uma vista de acordo com a distribuição física das máquinas, indicando quais os seus nomes e sistema operativo. Para outras, fará mais sentido organizar a informação por rede, seleccionando apenas um tipo de sistema operativo.

As vistas, além de facilitarem a interacção com o sistema de gestão, podem ser um importante mecanismo de controlo de acesso à informação de gestão. Isto porque as vistas podem restringir, não só os componentes que ficam visíveis ao utilizador de cada vista, mas também quais os atributos que podem ser acedidos.

Esta dissertação propõe um modelo de suporte à representação de redes de computadores destinada a ambientes de gestão de redes que toma os aspectos anteriores em consideração. O modelo proposto distingue duas componentes: a representação dos elementos da rede e a representação das múltiplas relações em que esses elementos podem estar envolvidos. A primeira é responsavel por caracterizar cada um dos elementos da rede de forma univoca, sejam eles sistemas de comunicações, sistemas computacionais ou aplicações. Na segunda, a vista, são agregadas as representações dos elementos da rede, de forma a apresentar uma determinada perspectiva coerente da rede alvo da gestão. A representação do comportamento da rede e dos seus elementos é conseguida graças a um mecanismo flexivel de propagação e tratamento de eventos.

A dissertação apresenta um protótipo do modelo proposto que se materializa numa extensão Tcl [51]. Esta extensão teve como base de trabalho a plataforma de gestão Tkined/Scotty. O sistema Scotty [60, 59] é um sistema de gestão de redes de domínio público que oferece uma interface para a linguagem Tcl [51]. A possibilidade de desenvolver aplicações de gestão personalizada usando uma linguagem interpretada como o Tcl, aliada ao facto de o Scotty se tratar de um sistema aberto, têm contribuído para a popularidade deste sistema. O protótipo aqui descrito aumenta o actual sistema Scotty com um sistema versátil de gestão de vistas.

A dissertação está estruturada do seguinte modo. No Capítulo 2 são apresentados os principais conceitos associados à gestão de rede e uma panorâmica sobre o trabalhos desenvolvidos nesta área. No Capitulo 3 são apresentados os trabalhos relevantes no suporte à representação de redes de computadores. O Capítulo seguinte descreve o nosso modelo de suporte à gestão de vistas que é concretizado e avaliado respectivamente nos Capítulos 5 e 6. O Capítulo 7 apresenta conclusões e direcções futuras.

Capítulo 2

Gestão de redes

Este capítulo apresenta uma panorâmica sobre os principais sistemas que abordam o problema da gestão de redes. Em primeiro lugar, apresentam-se os principais conceitos subjacentes à gestão de redes. Em segundo lugar, é dado ênfase aos diferentes paradigmas e arquitecturas que têm emergido nesta área. Finalmente, faz-se uma breve descrição de algumas plataformas de gestão.

2.1 O que é a Gestão de Rede?

Quando a utilização das redes de computadores estava circunscrita a pequenos grupos de investigação e possuía uma dimensão insignificante, comparada com a infra-estrutura que actualmente é utilizada por milhões de pessoas, não existia o conceito de gestão de rede. Quando ocorria um problema, os utilizadores que o detectavam executavam simples testes de actividade/conectividade para localizar a sua origem e, posteriormente, tomavam as medidas necessárias para a sua correcção (reiniciação do sistema, alterações na configuração, etc.). O crescimento das redes de computadores públicas e empresariais, tornou necessário criar e sistematizar mecanismos e procedimentos de

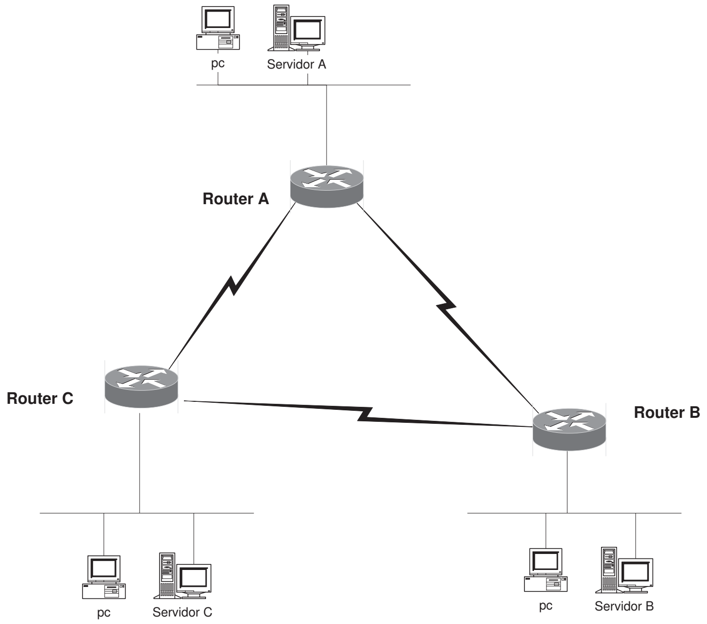

Figura 2.1: Cenário que ilustra a utilização de gestão de rede

gestão, dada a grande quantidade de componentes (maquinaria e programas) que entretanto foram sendo conectados.

A título de exemplo consideremos a pequena rede ilustrada na Figura 2.1. Mesmo neste exemplo simples, o administrador da rede pode obter uma enorme mais valia se utilizar ferramentas de gestão de rede. Vejamos os seguintes exemplos:

- Caso ocorra uma falha intermitente de uma interface ethernet do servidor A, o administrador da rede pode ser notificado de um incremento excessivo de tramas com erro na interface em causa, possibilitando a sua substituição atempada.

- A fiscalização [1] do tráfego em cada um dos segmentos de rede possibilita ao administrador da rede ajustar, por exemplo, a respectiva largura de banda optimizando desta forma os recursos disponíveis.

- A detecção de constantes alterações das tabelas de encaminhamento (route flap-

[1] Do Inglês, "monitoring".

ping) pode ser sinónimo de instabilidade no encaminhamento ou de encaminhadores (*routers*) mal configurados. Obviamente é preferível que seja o administrador da rede a descobrir os erros de configuração antes destes serem detectado pelos utilizadores.

- A fiscalização e tratamento da informação relativa à utilização da rede é um instrumento fundamental para o administrador da rede verificar o cumprimento dos níveis de serviço (*Service Level Agreements, SLA*) contratados ao operador de telecomunicações.

- A existência de determinado tipo de tráfego tal como pacotes com encaminhamento na origem (*source-route*) ou um excessivo número de tentativas de conexão para um mesmo sistema, poderá indicar uma tentativa de intrusão ou um ataque à rede. É claro o interesse do administrador em ser notificado destas ocorrências.

Os parágrafos anteriores vêm demonstrar que, quando se tenta definir área de gestão de rede, existe uma tendência para o fazer recorrendo à enumeração de um conjunto de exemplos que demonstram a sua utilidade. No entanto, parece-nos particularmente feliz a definição de gestão de rede apresentada por Uyless Black [6] :

> É actualmente aceite que a gestão envolve o planeamento, organização, fiscalização, registo e controlo de actividades e recursos. Esta definição pode ser certamente aplicada à gestão de rede. No entanto, a gestão de rede baseada no OSI ou na Internet encontra-se focada principalmente na fiscalização, registo e controlo das actividades e recursos da rede. Os restantes dois aspectos da gestão de rede, planeamento e organização, não se encontram abrangidos, mas são na realidade os aspectos mais importantes na gestão de rede e que consomem a maior parte dos recursos e talentos das organizações.

> De facto, se a rede não for devidamente planeada e organizada, são fúteos os recursos aplicados na sua fiscalização, registo e controlo.

A *International Standards Organization* (ISO) criou um modelo de gestão de rede que responde de uma forma estruturada aos cenários e definições anteriormente apresentados, através do estabelecimento de cinco áreas para a gestão de rede:

Gestão de Falhas que tem como objectivo registar, detectar e responder a condições de falha da rede.

Gestão de Desempenho que pretende quantificar, medir, analisar e controlar o desempenho da rede nas suas mais variadas vertentes (utilização, incidência de falhas, etc.).

Gestão da Configuração que pretende inventariar todos os recursos disponíveis na rede, manter um registo das respectivas configurações e um histórico das alterações.

Gestão de Utilização que pretende especificar, registar e controlar o acesso aos diferentes recursos da rede.

Gestão de Segurança que pretende controlar o acesso aos recursos da rede de acordo com uma política de segurança predefinida.

2.2 Infra-estrutura de Gestão de Rede

Vimos na secção anterior que a gestão da rede implica a interacção do seu administrador com os diferentes componentes, máquinas e programas, a ela ligados. Como os dispositivos se encontram distribuídos pela rede é necessário que o administrador tenha remotamente a capacidade de recolher dados ou alterar configurações. Para melhor explicar os diferentes elementos associados à gestão de rede, consideremos como exemplo a gestão de um banco com vários balcões:

O administrador do banco tem como missão garantir o bom funcionamento do banco. Para tal, recebe periodicamente dos balcões relatórios de gestão sob as mais diferentes formas: listagens, indicadores de actividade, etc. Além disso, pode ser notificado pelos gerentes dos balcões dos problemas para os quais estes não têm capacidade de decisão. O administrador pode ainda contactar os gerentes dos balcões para obter esclarecimentos ou dar instruções. Dependendo da dimensão do banco, o contacto entre o administrador e os gerentes pode ser directo ou indirecto através de chefias intermédias, podendo os gerentes ter maior ou menor autonomia de decisão.

Temos implícito neste cenário uma entidade gestora da organização (o administrador do banco), entidades geridas (os balcões representados pelo seu gerente), protocolos de comunicação (correio interno para a distribuição dos relatórios, telefone para os contactos entre o administrador e os gerentes) e dados (listagens e indicadores). A dimensão do banco reflecte o seu modelo de gestão (centralizado ou distribuído).

A arquitectura de uma infra-estrutura de gestão de rede, ilustrada na Figura 2.2, é conceptualmente idêntica a este modelo organizacional diferindo na terminologia:

- A **Entidade Gestora** (*Managing Entity*) é uma aplicação que corre numa estação de gestão de rede (*Network Management Station, NMS*). É responsável pela armazenamento, processamento e análise da informação de gestão.

- A **Entidade Gerida** (*Managed Device*) é uma componente da rede alvo da gestão. Corresponde no exemplo anterior ao balcão. A Entidade Gerida pode ser um componente físico tal como um servidor, um encaminhador, um *modem* ou um componente lógico tal como um serviço de rede (WWW por exemplo). Uma Entidade

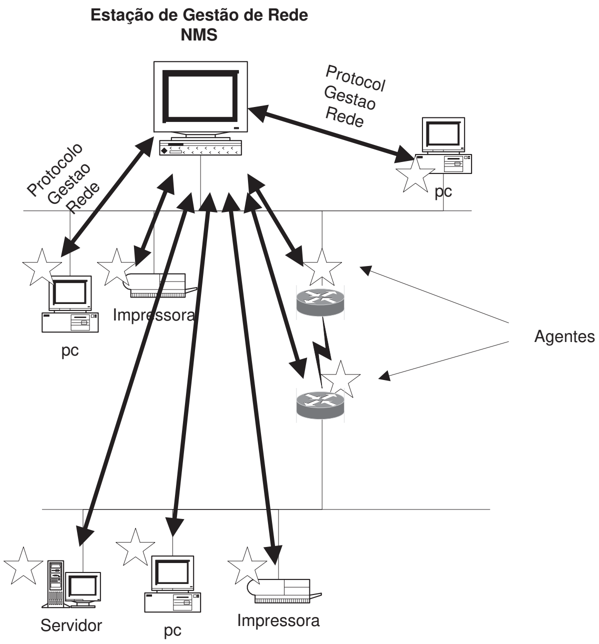

Figura 2.2: Principais componentes da arquitectura de gestão de rede

Gerida pode ser composta por vários objectos de gestão: de *hardware* como disco, interface *ethernet*; ou *software* tal como a configuração do equipamento. Os objectos de gestão têm associados vários atributos cujo conjunto forma a Base de Informação de Gestão (*Management Information Base*, *MIB*). O controlo da entidade gerida é efectuado através do seu agente (*Network Management Agent*) que corresponde ao gerente do exemplo anterior.

- O **Protocolo de Gestão** (*Network Management Protocol*) que suporta a relação entre a entidade de gestão e as entidades geridas.

2.3 Arquitecturas de gestão de rede

No exemplo da administração de um banco descrito na secção anterior, foi referido que o modelo de gestão estava relacionado com a dimensão do banco. De facto, o modelo centralizado (eficaz na gestão do banco se este tiver poucos balcões) torna-se incomportável para o seu administrador quando se verifica um aumento no número de balcões e consequentemente aumenta o processamento de listagens e indicadores. Neste caso o administrador do banco é obrigado a delegar as suas tarefas, distribuíndo desta forma a gestão do banco.

Como iremos ver nos próximos capítulos, algo semelhante se passou na Gestão de Rede: inicialmente o modelo centralizado foi concretizado com sucesso mas devido à sua falta de escalabilidade foi necessário encontrar modelos distribuídos de gestão.

Serão agora descritas aquelas que consideramos ser as principais propostas apresentadas por diferentes comunidades/organizações que abordam o problema da gestão de rede. Será patente a evolução de soluções, a nível da sua complexidade e do seu grau de distribuição.

2.3.1 IETF (Internet Engineering Task Force)

SNMP

Contrariamente ao que o nome pode sugerir, o SNMP (Simple Network Management Protocol) é mais do que um protocolo de transporte de informação entre a Estação de Gestão de Rede e os seus agentes. Na realidade o SNMP é um protocolo composto por:

- Definições dos objectos de gestão (*Network Management Objects*) também designados por objectos MIB (*MIB object*). Um objecto MIB (Management Information Base) pode ser um endereço IP, um contador, uma descrição. Os objectos MIB definem a informação de gestão que é mantida por um agente.

- Uma linguagem de definição dos dados, conhecida por SMI (*Structure of Management Information*) que define os tipos de dados, um modelo de objectos e regras para a escrita da informação de gestão. Os objectos MIB são especificados nesta linguagem.

- Um protocolo, SNMP, para o transporte da informação e comandos entre a entidade de gestão e os agentes

Neste protocolo, a principal motivação é a simplicidade do agente e a utilização optimizada dos recursos de rede. O protocolo SNMP foi desenhado tendo como premissa a sua utilização sobre o protocolo User Datagram Protocol (UDP) em detrimento de um protocolo orientado à conexão como o Transport Control Protocol (TCP). A redução do consumo de memória (característica dos protocolos que mantêm o estado das ligações) e de utilização do processador no agente, foram as principais motivações para esta decisão. Adicionalmente, sendo o SNMP baseado no modelo centralizado, a quantidade de conexões TCP que o servidor necessitaria de manter para dialogar com os agentes levantava questões de escalabilidade.

Uma mensagem SNMP é composta por um cabeçalho e por um campo de dados, tal como se ilustra na Figura 2.3. O cabeçalho é composto pela versão do protocolo e por um identificador designado por *comunidade*. A comunidade é um mecanismo primário de autenticação que se encontra disponível na primeira versão do SNMP: O agente SNMP verifica se a sua comunidade confere com aquela que se encontra inserida na mensagem SNMP. Existem cinco tipos de mensagens distintos no SNMPv1 [9]:

GetRequest: Mensagem enviada pela Estação de Gestão de Rede ao Agente para obter valores específicos da MIB do agente. O Agente responde com uma mensagem `GETResponse`.

GetNextRequest: Semelhante à primitiva anterior excepto no facto do objecto MIB retornado ser a instância imediatamente a seguir ao objecto MIB especificado. Esta

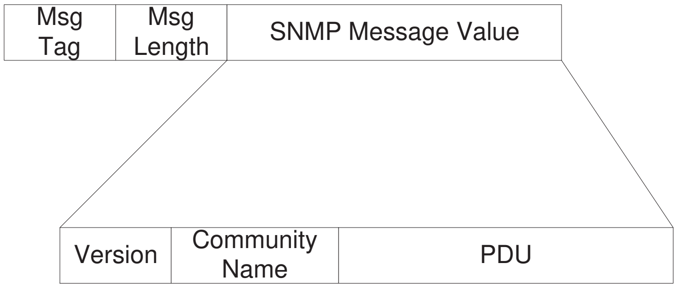

Figura 2.3: Estrutura de uma mensagem SNMP

primitiva permite à Estação de Gestão de Rede aceder a informação do agente sem saber exactamente quais os objectos suportados por este.

SetRequest: Esta primitiva é utilizada pela Estação de Gestão de Rede para alterar o valor de objectos MIB dos agentes que suportem esta operação.

GetResponse: É utilizado pelo agente para responder a pedidos de `GetRequest` ou `GetNextRequest`.

Trap: Esta primitiva é utilizada pelo agente para enviar notificações à Estação de Gestão de Rede quando ocorrem neste determinados eventos. Esta facilidade tem em vista a notificação da estação de gestão de um conjunto reduzido e pré-definido de eventos: `warmStart`, `coldStart`, `linkUp`, `linkDown` e `autheticationFailure`. Como o SNMP não suporta nenhum mecanismo de retransmissão, a estação de gestão terá, por sua iniciativa, de interrogar o agente (através de pedidos `GetRequest` ou `GetNextRequest`), caso seja necessário garantir a actualidade dos eventos que provocam a emissão de `traps`.

O fluxo das mensagens de SNMP é apresentado na Figura 2.4.

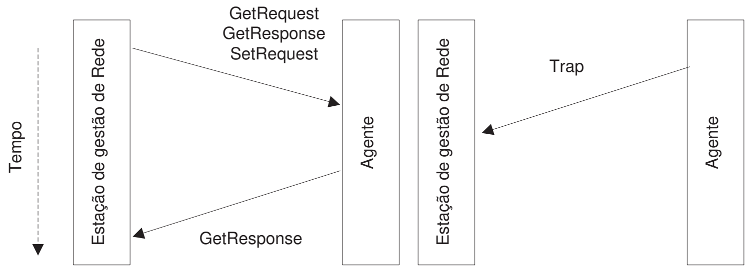

Figura 2.4: Comportamento do protocolo SNMP

SMI

O SMI (*Structure of Management Information*) é a linguagem usada na definição da informação de gestão que reside nos agentes. Esta linguagem é necessária para garantir que a sintaxe e a semântica dos dados sejam definidos sem ambiguidades. De notar que o SMI não define o conteúdo dos dados mas apenas o modo como essa informação é especificada. O SMI baseia-se por sua vez na linguagem ASN.1[ISO8824] (*OSI Abstract Syntax Notation*). Esta linguagem suporta a definição de tipos de dados simples e compostos. Um tipo de dados com especial importância é o OID (*object Identifier*) que define um objecto através de uma cadeira hierarquizada de inteiros não negativos. Os OID são registados pelas organizações responsáveis pela concretização de normalizações, tais como a ISO, ITU-T ou o IETF. O ASN.1 é utilizado nos ambientes de gestão OSI e SNMP para especificar a estrutura das mensagens dos respectivos protocolos de gestão e a estrutura da informação.

Na versão 1 do SNMP [55] são definidos três tipos básicos originários do ASN.1: `INTEGER`, `OCTECT STRING` e `OBJECT IDENTIFIER`; cinco tipos de dados aplicacionais: `Counter` (inteiro não negativo que incrementa monotonamente), `Gauge` (inteiro não negativo que pode incrementar ou decrementar), `TimeTicks` (inteiro não negativo que representa o tempo desde uma data preestabelecida em centésimos de segundo), `IpAddress` (endereço de rede) e `Opaque` (cadeia de caracteres não interpretada de

```
ospfIfIpAddress OBJECT-TYPE
    SYNTAX IpAddress
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
    "The IP address of thes OSPF interface"
    ::= { ospfIfEntry 1}
```

Figura 2.5: Exemplo da definição de um objecto MIB

ASN.1). A única estrutura de dados permitida são tabelas bidimensionais compostas, cujos elementos são de um dos tipos anteriormente definidos. Cada tabela é composta por linhas, onde cada linha é representada por uma sequência ("`SEQUENCE OF`") de um tipo dentro dos que foram anteriormente referidos.

Adicionalmente, o SMI define a sintaxe da linguagem para a especificação de objectos: a cláusula `OBJECT-TYPE` é utilizada para especificar a semântica, o estado e o tipo de objecto. Tem quatro cláusulas: `SYNTAX` (tipo de dados), `MAX-ACCESS` (especifica se o objecto é de leitura, escrita, se pode ser criado ou está incluído numa notificação), `STATUS` (indica se o objecto é valido, obsoleto ou desaprovado) e `DESCRIPTION` (descrição textual do objecto). Um exemplo é ilustrado pela Figura 2.5.

Como se pode observar, a sintaxe do SMI é simples e algo reduzida. Esta abordagem está mais uma vez relacionada com o objectivo de minimização dos recursos envolvidos na utilização do sistema, dado que a codificação e descodificação dos dados pode ser computacionalmente pesada. A utilização de uma sintaxe mínima torna possível a redução deste custo.

Num agente SNMP, a unidade de informação mais simples é o objecto, caracterizado por ser de um determinado tipo e por conter valores semanticamente de acordo com esse tipo. Quando o objecto é instanciado torna-se necessário identificá-lo de uma

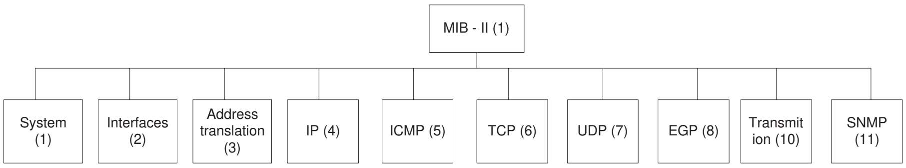

Figura 2.6: Estrutura da MIB II

forma unívoca. No SNMP é utilizada a convenção dos OID para referenciar as instâncias. Cada definição de objecto é caracterizada por um OID registado e universal. Quando um objecto é instanciado passa a ser identificado por um nome composto pelo OID do objecto ao qual é adicionado um sufixo que identifica universalmente a instância. Por exemplo o objecto `udpInErrors` (`1.3.6.1.2.1.7.3`) pode ter a instância `1.3.6.1.2.1.7.3.0`.

MIB

Nos agentes, as diferentes instâncias dos objectos encontram-se armazenados e estruturados em MIBs: uma colecção estruturada e classificada das definições dos diferentes objectos que no seu todo caracterizam o dispositivo ou elemento alvo da gestão. Em 1988 foi apresentada a primeira MIB contendo um conjunto de objectos relativamente reduzido. Verificou-se a sua expansão em 1990, data da apresentação da MIB II. A MIB II [56] contém já 10 grupos hierárquicos e 171 objectos conforme representado na Figura 2.6. Adicionalmente os fabricantes de equipamentos de rede foram disponibilizando MIB específicas que descrevem melhor cada um dos seus componentes.

O SNMPv1 teve um enorme sucesso e recebeu uma larga aceitação em poucos anos: em 1993, cada dispositivo existente no mundo IP tinha um agente SNMP embebido. O sucesso do SNMPv1 deveu-se essencialmente à simplicidade do modelo e ao facto de potenciar a integração da gestão de sistema heterogéneos.

No entanto, o crescimento das redes IP ao longo dos anos 90, o aumento do número de dispositivos e serviços a gerir e o aumento da dimensão da MIB, expuseram a maior fraqueza do modelo centralizado em que se baseava o SNMPv1: a sua escalabilidade. Tornou-se necessário encontrar um novo paradigma.

SNMPv2

O SNMP versão 2 (SNMPv2) foi apresentado em 1993, não tendo colhido uma grande aceitação por parte dos utilizadores (fabricantes e clientes) devido à complexidade do sistema e da sua componente de segurança. Em 1994 do grupo de trabalho do SNMPv2 iniciou um trabalho de revisão que resultou em 1996 na apresentação de três versões diferentes do SNMPv2 [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]:

- SNMPv2u: Versão que inclui componentes de segurança baseada em utilizadores
- SNMPv2* (SNMPv2star): uma abordagem experimental que combina o SNMPv2 original com o SNMPV2u
- SNMPv2c: Baseado no SNMPv2 original ao qual foram retiradas as componentes de segurança.

No essencial a expansão do SNMPv1 para o SNMPv2 verificou-se em três áreas:

- O SMI do SNMPv2 expande o SMI do SNMPv1:
  - Suporta novos tipos de dados nomeadamente inteiros de 64 dígitos e endereços NSAP do OSI.
  - Incorpora uma nova convenção para a criação e remoção de linhas nas tabelas. Cada tabela contém um novo objecto, `RowStatus`, cujo estado indica a operação que se encontra em processamento na tabela.

  - Introduz o conceito de módulos (*MIB modules*) que permite agrupar MIBs relacionadas.

- Foi disponibilizada uma MIB, M2M MIB (*Manager-to-Manager MIB*), para o suporte a uma arquitectura de gestão hierarquica. Esta MIB possibilita a existência de gestores intermédios que a utilizam para informar a hierarquia superior das actividades dos seus respectivos agentes. Esta Mib é composta por dois grupos:

  - `Alarm Group`: Define um conjunto de objectos para a descrição e configuração de alarmes numa entidade SNMPv2, por forma a que esta possa agir como gestor e como agente.

  - `Event Group:` Define um conjunto de objectos necessários à descrição e configuração de eventos numa entidade SNMPv2.

- Ao nível do protocolo são adicionadas duas novas primitivas `GetBulkRequest` e `InformRequest`.

  - `GetBulkRequest` permite a transferência de grandes quantidades de dados sem que seja necessário transferir um campo de cada vez, tal como acontece com a primitiva GetNextRequest.

  - O `InformRequest` possibilita a existência de um mecanismo semelhante às notificações(`traps`), entre estações de gestão de rede, possibilitando a constituição de um modelo hierárquico de gestão de rede.

- A nível da segurança foram adicionados ao SNMP um conjunto de serviços:

  - *Verificação da integridade dos dados*: Através de um algoritmo de síntese (MD5) é calculada uma síntese de 128 dígitos de uma parte da mensagem SNMP.

Este valor é incluído na mensagem por forma a que o destinatário possa verificar a coerência da mensagem. Foi ainda incluído na mensagem uma estampilha temporal que permite ao destinatário verificar por um lado a idade da mensagem e por outro lado a sua sequência.

– *Autenticação da origem dos dados*: A síntese anteriormente referida é calculada com base em parte da mensagem SNMP e numa chave que é do conhecimento prévio dos interlocutores.

– *Confidencialidade dos dados:* É utilizado o algoritmo criptográfico simétrico DES (*Data-Encryption Standard*) na cifra das mensagens.

SNMPv3

O SNMP versão 3 [21, 70, 10, 36, 7, 71], entretanto concluído (Abril de 1999), endereça algumas das deficiências existentes no SNMPv2 com especial destaque para as componentes de segurança e administração. Relativamente às versões 1 e 2 do SNMP, a versão 3 disponibiliza as seguintes facilidades:

- Identificação das entidades SNMP para proporcionar a comunicações somente entre entidades SNMP conhecidas. Cada entidade SNMP tem um identificador - *SnmpEngineD* - que é incluído em cada mensagem. Assim a comunicações entre entidades SNMP só é possível quando ambos os interlocutores conhecem o *SnmpEngineD* do seu correspondente. É excepção a esta regra o caso das notificações.

- Definição de políticas de segurança que se materializam no serviço de autenticação de mensagens pela protecção contra:

  – Modificação da informação - Protecção contra tentativas de alteração de mensagens SNMP em transito através da verificação da integridade dos dados.

- Falsos administradores - Protecção contra tentativas de execução de operações de gestão por parte de entidades não autorizadas para tal através da autenticação do emissor dos dados.
- Alteração dos canais estabelecidos entre entidades SNMP - Protecção contra tentativas de reordenação, atraso ou repetição de mensagens com o objectivo de se conseguir concretizar operações de gestão não autorizadas.
- Violação da confidencialidade dos dados - Protecção contra tentativas de obtenção da informação trocada entre entidades SNMP.

- Especificação do Modelo de Segurança definido pelo utilizador (*User-based Security Model - USM*). Este modelo caracteriza os diferentes tipos de comunicação possíveis entre entidades SNMP:
  - Comunicação sem autenticação e privacidade (NoAuthNoPriv)
  - Comunicação com autenticação e sem privacidade (AuthNoPriv)
  - Comunicação com autenticação e privacidade(AuthPriv)
- Ambiente de trabalho (*Framework*) destinado à definição dos algoritmos de autenticação e privacidade. Actualmente são suportados os algoritmos MD5 (Message Digest Algorithm number 5) e SHA (Secure Hash Algorithm). O Algoritmo DES (Data Encryptation Standard) é o único algoritmo suportado pelo USM para a comunicação com privacidade.
- Definição de um procedimento de busca do SnmpEngineID de uma entidade SNMP.
- Definição de um procedimento de sincronização de relógios entre entidades SNMP.

O SNMPv3 ao resolver, de uma forma sólida e coerente, as deficiências de segurança existentes nas versões anteriores, marca uma importante posição de destaque nos protocolos de gestão de rede dada a importância desta faceta na gestão de sistemas.

RMON

As sondas RMON (*Remote Network Monitoring*) [66, 67], dispositivos remotos dedicados à fiscalização da rede, foram o primeira forma de hierarquização adicionada ao SNMPv1.

Estas sondas têm como função fiscalizar a rede, com especial destaque para segmentos *ethernet*, através da recolha de estatísticas. Podem tomar a forma de dispositivos específicos que são ligados à rede ou podem simplesmente corresponder a programas que adicionam funcionalidades de fiscalização a dispositivos de rede (encaminhadores, comutadores, etc). Os dispositivos físicos são normalmente pequenos computadores dotados de uma placa de rede e muita memória (necessária ao armazenamento da informação capturada). Estes equipamentos podem armazenar informação relativa a vários dias, semanas ou meses, dependendo obviamente da memória disponível e da sua configuração (intervalos de amostragem, número de variáveis a recolher, etc).

A adopção do RMON possibilitou que as estações de gestão de rede pudessem delegar um conjunto de tarefas de fiscalização e notificação, reduzindo por um lado os recursos computacionais necessários e por outro lado o trafego de informação de gestão. Além disso, a ocorrência de uma partição entre a estação de gestão de rede e a sonda RMON não impede que esta última continue a recolher estatísticas.

As actividades realizadas pelo RMON são estáticas, uma vez que somente aquelas que foram definidas pelo fabricante do equipamento é que podem ser utilizadas.

2.3.2 ISO

A arquitectura OSI fornece um conjunto de recursos que permitem a comunicação entre sistemas abertos de acordo com os seus serviços e protocolos. A gestão e controlo das actividades de comunicação e dos recursos nela envolvidos é proporcionado pelo ambiente de gestão OSI que disponibiliza as ferramentas e os serviços necessários às actividades de gestão.

Os recursos são representados no ambiente de gestão OSI por entidades designadas por "Objectos Geridos" (*Managed Objects, MO* [50, 48, 49]). Um MO é uma representação abstracta de entidades físicas (encaminhador, *hub*, etc) ou de entidades lógicas (protocolo de comunicação, aplicação, etc). É caracterizado por um conjunto de atributos que definem as sua propriedades, um conjunto de operações (métodos) que caracterizam o seu comportamento e um conjunto de eventos (notificações) que o *MO* emite assincronamente. Um *MO* é membro de determinada classe de objectos que caracteriza todos os *MO* que detêm os mesmos atributos, as mesmas operações e os mesmos eventos. Os MO são descritos na notação GDMO (*Guidelines for Definition of Managed Objects*). O GDMO é uma linguagem formal, destinada a definir e modelar a informação de gestão que caracteriza os *MO*.

O modelo de gestão OSI obedece ao paradigma das arquitecturas orientadas a objectos fazendo por isso uso a conceitos como a herança e inclusão. As classes de objectos geridos (*MOC, Managed Object Classes*) podem ser especializadas em subclasses que herdam e estendem as características das superclasses, desde novos atributos, a novas acções ou novas notificações. Um *MO* pode incluir outros MO que por sua vez incluem outros *MO*.

O conjunto das classes de objectos geridos e suas respectivas relações (herança, inclusão) encontram-se definidos e armazenados na MIB OSI. Contrariamente às linguagens orientadas a objectos, a unidade de reutilização não é a classe mas o pacote (*package*). Um pacote é um conjunto de atributos, acções, notificações e respectivo comportamento. As MOC caracterizam-se por conterem um conjunto de pacotes obrigatórios podendo ou não conter pacotes opcionais.

O paradigma da gestão OSI caracteriza-se pela existência de dois processos distintos: o processo gestor (*manager*) e o processo agente (*agent*) O agente manipula os MO executando directamente operações solicitadas pelos gestores. Os agentes são ainda responsáveis pelo envio de notificações aos gestores.

A arquitectura de gestão OSI tanto suporta o modelo centralizado como o modelo hierárquico. No modelo centralizado, uma única Estação de Gestão de Rede é responsável pelo processamento pela recolha e processamento da informação de gestão. No modelo hierárquico é possível a convivência de várias Estações de Gestão de Rede que cobrem diferentes áreas geográficas que se pretende gerir.

O sistema de gestão do OSI define ainda os mecanismos necessários à fiscalização, controlo e coordenação dos MO. As comunicações de gestão são da responsabilidade do SMAE (*System Management Aplication Entity*) que define os serviços que são oferecidos pelo sistema de gestão OSI. Os SMAEs enviam os pedidos e recebem as notificações. São compostos por um ou mais ASE (*Aplication Service Entity*): SMASE (*System Management Aplication Service Element*), CMISE (*Common Management Information Service Element*), ROSE [43, 44] (*Remote Operation Service Element*) e ACSE (*Association Control Service Element*) [47]. O SMASE [50] é responsavel pela identificação e definição dos aspectos relativos à informação que deve ser trocada entre sistemas. O CMISE suporta a partilha de informação de gestão entre sistemas envolvidos na gestão OSI. O CMISE é por sua vez composto pelo CMIS [46] (*Common Management Information Service*) e pelo CMIP [45] (*Common Management Information Protocol*). Na Figura 2.7 é descrita a arquitectura do framework de gestão OSI.

O CMIS é um serviço utilizado pela arquitectura de gestão OSI para tratar das comunicações relacionadas com a gestão dos MO. Define um conjunto de primitivas, respectivos parâmetros e toda a informação necessária à descrição semântica de cada primitiva. O CMIS disponibiliza sete primitivas, cujas interacções são ilustradas na Figura 2.8, nomeadamente: M-GET, para recolher valores dos atributos de um, ou mais MO; M-SET,

Figura 2.7: Arquitectura CMIP

para alterar valores dos atributos de um ou mais MO; M-CREATE, para criar novos MO; M-DELETE, para eliminar MO; M-EVENT-REPORT, para notificar o gestor da ocorrência de um evento; M-CANCEL-GET, para cancelar o pedido de M-GET; M-ACTION, para executar uma acção num MO.

As operações de `GET`, `SET`, `ACTION` e `DELETE` podem ser executadas sobre múltiplos objectos com um único pedido. Este pedido é expandido no agente conforme as parâmetros de âmbito (*scoping*), filtragem ou sincronização definidos. Uma vez que os MO são referenciados de acordo com as suas relações de inclusão e organizados conforme a árvore da MIB, é possível o envio de um pedido CMIS para um objecto e seleccionar os objectos nele incluídos que obedeçam a um determinado âmbito. Adicionalmente, através dos mecanismos de filtragem, é possível seleccionar os objectos baseado em asserções dos valores dos atributos combinado com operações booleanas. O âmbito e a filtragem são dois mecanismos poderosos que fornecem aos agentes OSI as funcionalidades características de uma base de dados orientada a objectos. Como resultado da aplicação destes mecanismos, a lógica das aplicações gestoras pode ser simplificada, reduzindo-se ainda de uma forma substancial o tráfego de gestão. Quando é solicitada uma operação sobre múltiplos objectos que obedeçam a um determinado âmbito e

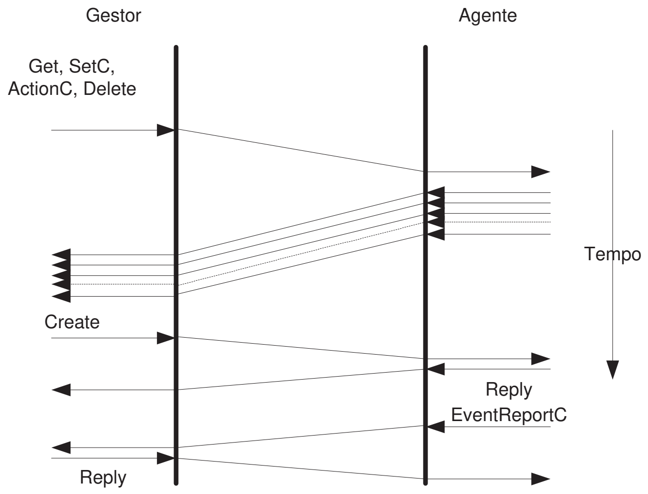

Figura 2.8: Interacções do CMIS

filtragem, é possivel garantir a atomicidade da operação.

O CMIP especifica o protocolo utilizado pelo CMIS e pelos MO na troca de informação de gestão e define os procedimentos necessários à troca de informação entre aplicações. O CMIS/CMIP pode ser visto com o protocolo de execução remota, baseado em mensagens assíncronas, tendo sido desenhado para se executar fundamentalmente sobre a pilha de protocolos do OSI. Embora o CMIP possa ser disponibilizado sobre um leque bastante variado de protocolos de nível de transporte e de rede, as concretizações sobre outras familias de protocolos, nomeadamente o TCP/IP [54], nunca tiveram grande penetração no mercado. A arquitectura do CMOT, CMIP sobre TCP/IP [68], encontra-se descrita na Figura 2.9.

Embora a arquitectura de gestão OSI não tenha tido sucesso na comunidade Internet e nos ambientes de gestão de sistemas de informação, obteve um enorme sucesso na gestão das redes de telecomunicações. O TMN (Telecomunications Management Network) é uma arquitectura funcional, que assenta nos protocolos de gestão OSI, e que suporta um conjunto vasto de funções (configuração, faltas, desempenho, segurança e controlo de utilização) sobre diferentes perspectivas (rede, serviço, gestão do negócio, etc).

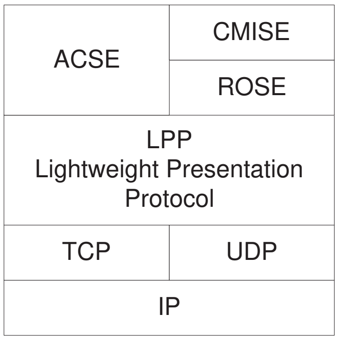

Figura 2.9: Arquitectura CMOT

2.3.3 OMG (Object Management Group)

Enquanto os modelos de gestão propostos pelo IETF e pelo OSI normalizam as interfaces das aplicações gestoras e agentes, a proposta da OMG tem como objectivo disponibilizar uma interface de comunicação entre objectos baseado na arquitectura CORBA (Common Object Request Broker Architecture) [32].

No modelo CORBA [31, 30], um objecto é uma entidade identificável e encapsulada que disponibiliza serviços. Os objectos encapsulam estado (atributos) e comportamento (operações) que podem ser acedidos através de uma interface que descreve o conjunto de operações passíveis de ser executadas pelo objecto. A interface de um objecto é especificado através da linguagem OMG IDL (Interface Description Language). O IDL, tal como o ASN.1, não corresponde a uma linguagem de programação, sendo utilizado apenas para descrever a interface dos objectos, cuja funcionalidade deverá ser posteriormente codificada usando uma linguagem de programação (C++, FORTRAN, etc). A Figura 2.10 seguinte descreve a concretização dos objectos.

O compilador de IDL compila o ficheiro IDL que contém a descrição das interfaces de um conjunto de objectos. Como resultado, o repositório de interfaces (repositório persistente das definições das interfaces dos objectos) é actualizado e são gerados rotinas de adaptação na linguagem de programação adoptada (por exemplo C++). O uti-

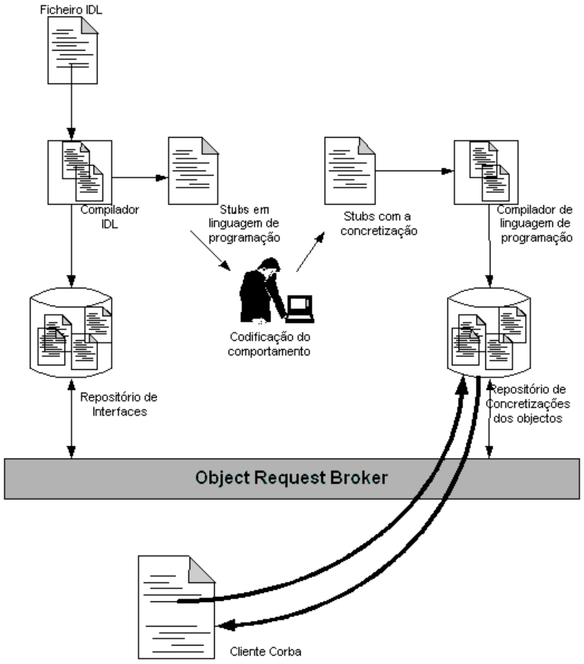

Figura 2.10: Funcionamento do CORBA

lizador terá de codificar o comportamento que pretende ver associado a cada método. Da compilação das rotinas de adaptação resultam normalmente bibliotecas partilhadas (*shared libraries*) cuja localização é adicionada a uma base de dados que contém as concretizações dos objectos (*Implementation Repository*).

Cada sistema que contenha instâncias CORBA, tem activo um serviço denominado ORB (Object Request Broker) responsável por, dado um determinado pedido de um cliente, localizar o objecto e a respectiva concretização, activar a execução do pedido e devolver o resultado (caso exista) ao cliente. O ORB é ainda responsável pela criação e eliminação de objectos.

A arquitectura CORBA foi projectada para ser uma infra-estrutura de suporte ao desenvolvimento e execução de aplicações distribuídas, em que a componente de comunicação poderia ser proprietária. Daí que a primeira versão (OMG 1.x) do CORBA não contemplasse um protocolo de acesso. Na versão 2.0 foi especificado um protocolo de invocação remota designado por GIOP (General Inter-Operability Protocol) que é suportado por dois protocolos de transporte: IIOP (Internet Inter-Operability protocol) para o mundo Internet e o D-CIOP (DEC Common Inter-Operability protocol).

A utilização de uma arquitectura orientada a objectos em actividades de gestão de rede manifestou-se em algumas propostas donde se destacam o TINA e o XoJIDM. O CORBA foi adoptado na arquitectura TINA [5] (Telecomunication Information Network Arquitecture) para o desenvolvimento de serviços e aplicações de gestão vocacionadas para o mercado das telecomunicações. O XoJIDM (X/Open Join Inter-Domain Management) é um grupo patrocinado pelo X/Open e pelo NMF (Network Management Forum) com o objectivo de criar ferramentas de interoperabilidade entre os mundos CMIP, SNMP e CORBA. Este grupo desenvolveu algoritmos de associação entre GDMO/ASN.1 e o IDL de CORBA [40] e entre as MIBs de SNMP e o IDL de CORBA [39]. Estas associações permitem aos programadores de CORBA escreverem gestores baseados em CMIP ou SNMP sem terem conhecimentos de GDMO, ASN.1 e CMIP, ou SNMP. Do mesmo modo, os programadores de GDMO, CMIS e SNMP também podem aceder a recursos, serviços ou aplicações construídos a partir de IDL.

2.3.4 DMTF (Desktop/Distributed Management Task Force)

O SNMP e o CMIP suplantaram a maioria das soluções de gestão proprietárias, nas áreas onde os sistemas abertos tiveram sucesso, nomeadamente as áreas da Internet e das Telecomunicações. No entanto, na área dos computadores pessoais, continuam a predominar as soluções de gestão proprietárias. O Novell ManageWise e o Microsoft SMS são bons exemplos disso.

Na segunda metade dos anos 90, a Desktop Management Task Force apresentou a especificação DMI [25] (Desktop Management Interface) com o objectivo de promover a normalização na área dos computadores pessoais. O desenho do DMI teve como motivações: ser independente de qualquer sistema computacional ou ambiente; ser independente de qualquer protocolo de gestão; de fácil adopção por parte dos fabricantes; utilizável localmente, sem necessidade de suporte de rede;de fácil associação aos protocolos de gestão vigentes (CMIP, SNMP). O DMI especifica quatro componentes:

Management Information Format (MIF): formato para a descrição da informação

Component Interface (CI): Formato utilizado para a transferência de dados. Utilizado pelos fabricantes para descreverem a informação de gestão do componente por forma a permitir a sua gestão.

Management Interface (MI): Mecanismo utilizado para a transferência dos dados. É utilizado pelas aplicações que pretendem gerir o componente.

Service Layer (SL): Conjunto de serviços que suportam a comunicação.

Para abordar esta área a Microsoft liderou um consórcio para a constituição do WBEM [62] (Web Based Enterprise Management). Esta proposta é constituída por um novo modelo de objectos, HMMS (HyperMedia Management Schema), um novo protocolo, HMMP (HyperMedia Management Protocol) e um novo ambiente de gestão dos elementos alvo, HyperMedia Object Manager (HMOM). No entanto, esta especificação teve pouco sucesso, uma vez que a adesão dos fabricantes a esta norma colocaria em causa o seu investimento nas plataformas de gestão proprietárias.

Actualmente o DMTF, cuja designação entretanto passou a significar Distributed Management Task Force, enveredou por uma tentativa de agrupar no WBEM um conjunto de tecnologias nomeadamente o CIM [26] (Common Information Model), o XML (Extensible Markup Language) e o HTTP (Hypertext Transfer Protocol). O CIM é utilizado para modelar os elementos da rede. O diálogo Gestor/Agente é efectuado segundo a especificação XML através do protocolo HTTP.

O CIM é um método de descrição da informação de gestão seja ela de rede, sistemas ou mesmo de gestão genérica de sistemas de informação. O CIM é uma linguagem de especificação orientada a objectos, baseada em UML (Unified Modeling Language) [1].

2.3.5 SUN/JAVA

Embora se tenha assistido na primeira metade dos anos 90 a um grande esforço na normalização dos modelos de gestão dos sistemas de informação, que se materializou no aparecimento de um vasto conjunto de soluções a nível de protocolos, modelos de informação e arquitecturas de gestão, a realidade mostra que as empresas na sua grande maioria adopta as ferramentas de gestão fornecidas pelos fabricantes dos equipamentos. Estas soluções apresentam como vantagens o facto de estarem perfeitamente adaptadas ao dispositivo que se pretende gerir. No entanto, apresentam algumas desvantagens importantes:

- Interfaces com utilizador não normalizados: se um administrador de rede gerir um conjunto heterogéneo de equipamentos é obrigado a aprender e a utilizar aplicações de gestão heterogéneas.

- Incoerência de versões: o principal objectivo de um fabricante de dispositivos de rede é o fornecimento de soluções de rede. A componente de gestão é normalmente relegada para segundo plano, sendo usual que a disponibilização de um dispositivo de rede ou de uma nova versão de software não seja acompanhada pela correspondente actualização da aplicação de gestão.

- Suporte em múltiplas plataformas: os fabricantes de dispositivos são obrigados a desenvolver e a suportar as suas aplicações em múltiplas plataformas (ambientes operativos, arquitecturas de gestão) com os inerentes custos que se reflectem no fabricante e no no cliente.

Para fazer face a este conjunto de problemas, surgiram algumas propostas de utilização de navegadores WEB para a gestão dos diferentes componentes de rede. Para isso, torna-se necessário que os fabricantes incorporem nos seus equipamentos o software de gestão sob a forma de *applets* (*embedded management applications*) [69]. Estas *applets*, escritas na linguagem Java, disponibilizam uma interface com o utilizador idêntico ao que é proporcionado pelas aplicações de gestão especificas, permitido ao administrador da rede a gestão do dispositivo através de RMI (Java Remote Method Invocation), em vez de SNMP. Os dispositivos de rede devem incorporar um servidor de *http* para suportar o dialogo entre o *browser* e o dispositivo. A Sun disponibilizou um conjunto de ferramentas e directrizes para a construção destas applets, cuja arquitectura tomou o nome de JMAPI (Java Management API Arquitecture) [64].

2.3.6 Outras propostas

A relação da gestão de rede com os sistemas de aplicações distribuídas encontra-se associada ao conceito de código móvel ou seja à possibilidade de incorporação dinamica nos agentes de novos procedimentos ou facilidades, flexibilizando as operações que podem ser executadas nos agentes e permitindo a delegação de competências.

O conceito de código móvel aparece inicialmente associado ao desenvolvimento de aplicações distribuídas, estando intimamente ligado a uma nova família de linguagens tais como o Telescript [37], Tycoon [38], Agent TCL [29], Java [28], TACOMA [34], etc [24]. Segundo Carzaniga *et al.* [8] existem três categorias de aplicações distribuídas baseadas em código móvel:

- Execução Remota (*Remote Evaluation, REV*): O cliente ao invocar um serviço de um servidor envia-lhe o nome do serviço, parâmetros e também o respectivo código do serviço. É exemplo desta categoria o sistema REV [63], que pode ser visto como uma extensão do comando Unix `rsh`.

- Código a pedido (*Code on Demand, COD*): Um cliente quando pretente executar uma determinada tarefa, recolhe de um servidor o código necessário à execução
- *Agente Móvel* (Mobile Agent, MA): Nesta caso, a unidade de execução tem a capacidade de migrar autonomamente de um sistema para outro, continuando a sua execução. Ou seja o agente possui o código mas não os recursos para a sua execução.

Existem várias propostas de utilização de código móvel na gestão de rede destacando-se os trabalhos de Gestão Delegada (*Management by Delegation, MbD*) [27] e de Redes Activas [65] (*Active Networks*). A Gestão Delegada tem como objectivo resolver os problemas do modelo de gestão centralizada através da movimentação do código ou das funções, para onde se encontram os dados. O principio de funcionamento é bastante simples:

O cliente envia o programa (*delegated agent*) para o servidor (*elastic server*), utilizando um protocolo designado por RDP (Remote Delegation Protocol). O servidor integra o programa e pode executá-lo de imediato ou agendar a sua execução. Isto é possível porque a arquitectura do servidor se baseia num ambiente de execução com várias actividades (*multi-threaded*) que partilham um mesmo espaço de endereçamento. Os processos que se executam neste ambiente são designados processos elásticos (*elastic processes*). Um processo elástico é uma encarnação de um programa que pode ser modificado, estendido ou contraido durante a sua execução. Não existe um formato fixo para estes agentes delegados: podem ser scripts, programas binários, etc.

O protocolo RDP permite transferir o código do agente do cliente para o servidor elástico e o respectivo controlo remoto da sua execução (é possível executar, agendar, suspender, cancelar a execução de determinado agente num servidor elástico). Esta facilidade aumenta as capacidades dos dispositivos de rede: se o contacto entre a estação de gestão de rede e o agente se perder, este último tem a capacidade de activar programas locais, previamente carregados, que permitam tomar medidas correctivas para eliminar ou minimizar o problema.

Finalmente, justifica-se mencionar o conceito de rede activas: permitir a incorporação de programas nos diferentes dispositivos da rede com a capacidade de manipular o fluxo dos dados na rede e de alterarem os dados. Esta ideia, algo revolucionária, infringe o principio de que a rede deve transportar os dados de uma forma opaca. Esta proposta for apresentada por Tennenhouse e Wetherall [65], tendo sido concretizada por Yemini [72].

Existem essencialmente duas abordagens ao conceito de redes activas: "Programable switch Appproach" [72] e "Capsule Approach" [65]. Na primeira abordagem o formato dos dados não é alterado sendo criado o mecanismo de distribuição de programas pelos diferentes elementos da rede que alteram o seu comportamento. A segunda abordagem considera que os pacotes de dados são programas encapsulados que são executados nos diferentes dispositivos da rede por onde passam.

2.4 Plataformas de gestão de rede

2.4.1 Plataformas do domínio público

Este capítulo apresenta algumas das propostas de ferramentas de domínio publico para a gestão de sistemas e redes. Grande parte delas são distribuídas gratuitamente através da Internet.

Ferramentas baseadas no OSI

O OSIMIS é uma plataforma de gestão orientada a objectos baseada no modelo OSI, e concretizada em C++. Disponibiliza um ambiente de desenvolvimento de aplicações de gestão que esconde os pormenores dos serviços de gestão OSI através de um conjunto de interfaces, possibilitando que o programador se concentre no desenho de aplicações de gestão.

Ferramentas baseadas no SNMP

CMU A universidade de CMU desenvolveu um pacote de programas para o suporte a SNMP, composto por uma biblioteca de desenvolvimento, um conjunto de utilitários e interfaces para Tcl/Tk. Os utilitários tomam a forma de programas, passíveis de serem executados através de linha de comando, que suportam as actividades usuais do SNMP.

Scotty/Tkined As dificuldades encontradas no desenvolvimento de aplicações de gestão utilizando tecnologias baseadas na compilação estática (ex: linguagem C) ou através de geração automática de programas, levaram alguns grupos de investigação a procurar novas formas de desenvolvimento de sistemas de gestão de redes e sistemas. O desenvolvimento rápido de aplicações, a facilidade de utilização e a portabilidade são os principais requisitos da nova plataforma de desenvolvimento de aplicações de gestão. Uma abordagem interessante baseou-se na adopção de linguagens de "*scripting*", donde se destacam o Perl e o Tcl (Tool Command Language).

O Tnm/Scotty é uma extensão para o Tcl que disponibiliza uma interface de acesso a um conjunto variado de protocolos de rede, possibilitando o desenvolvimento rápido e simples de *scripts* destinados à gestão de redes e sistemas.

Adicionalmente, o Scotty disponibiliza um conjunto de utilitários que simplificam o trabalho dos programadores. Destacam-se o `Tnm::Job`, que permite activar procedimentos periodicamente; `Tnm::syslog`, que permite escrever no histórico do sistema; `Tnm:netdb`, que dá acesso às bases de dados de rede locais (`hosts`, `networks`, etc);etc.

O Scotty está intimamente ligado ao Tkined, um editor gráfico destinado à criação e manutenção de representações da rede. O Tkined funciona como interface gráfica para um conjunto de aplicações que normalmente actuam sobre a representação da rede. Estas aplicações correm em processos distintos e comunicam com o Tkined através de um protocolo cuja interface no Scotty se encontra encapsulado no comando `Tnm:ined`.

2.4.2 Plataformas comerciais

Como vimos anteriormente, a gestão de rede segundo a ISO engloba diferentes disciplinas de gestão: gestão de falhas, gestão de desempenho, gestão de configuração, gestão de utilização e gestão de segurança. As actuais soluções comerciais de gestão de sistemas de informação, onde se engloba a gestão de redes, reflectem a tendência de mercado que se verificou na segunda metade da década de 90, de englobar numa mesma plataforma a gestão dos diferentes componentes do sistema. Na verdade estamos perante um objectivo muito interessante e ambicioso: fornecer às empresas uma plataforma que lhes permita gerir as suas redes e respectivo equipamento activo, as suas aplicações, as infra-estruturas de suporte ao sistema (ar condicionado, electricidade, etc.) e realizar também a gestão dos processos. Idealmente esta plataforma deverá permitir correlacionar as tecnologias com as actividades dos seus utilizadores. A falha de um encaminhador terá pouco interesse para os accionistas. No entanto, a quebra de produtividade derivada da falha do encaminhador adquire outra importância junto deles. É baseado nestas preocupações que os fabricantes de plataformas de gestão de sistemas de informação incluíram novas funções nos seus produtos:

1. **Help-Desk:** Componente que trata da gestão dos problemas. Tipicamente inclui

mecanismos que possibilitam a abertura e registo de problemas, a sua distribuição para as pessoas adequadas à sua resolução, o endereçamento a níveis hierárquicos superiores caso a resolução do problema ultrapasse o nível de serviço acordado entre o cliente e o fornecedor (*Service Level Agreement, SLA*). Inclui ainda a possibilidade do cliente poder verificar a evolução do processo de resolução do seu problema.

2. **Controlo Remoto:** Componente que permite ao administrador ter o controlo da estação de trabalho do cliente. Muitos problemas apresentados pelos utilizadores podem ser resolvidos remotamente através da intervenção directa sobre o ambiente do utilizador. Normalmente as funções de controlo remoto incluem as capacidades de transferência de ficheiros e de inicialização do sistema.

3. **Inventário de Software:** Componente que permite manter um inventário dos sistemas operativos e sistemas aplicacionais existentes nos diferentes sistemas alvo da gestão proporcionando a obtenção da informação necessária à realização de actualizações ou alterações de configuração, a verificação de eventuais instalações de software não autorizado, etc. O inventário é uma peça fundamental de uma das principais disciplinas da gestão de sistemas de informação: a gestão da mudança (*Change Management*).

4. **Gestão de Licenças:** Componente que permite ao administrador estabelecer o número de licenças necessárias das aplicações para suportar um determinado conjunto de utilizadores. Este conhecimento permite uma redução considerável de custos, uma vez que torna possível a aquisição de um número de licenças exacto em vez de se efectuar uma aquisição por estimativa.

5. **Distribuição de Software:** Componente que permite a distribuição de novas aplicações, correcções de erros e actualizações de programas de uma forma automáti-

ca sem necessidade de intervenção local, de um técnico especializado, reduzindo significativamente o tempo e custos necessários à instalação de cada distribuição de software.

6. **Descoberta dos recursos da rede:** Componente que tem como função descobrir automaticamente todos os recursos computacionais que se encontram ligados à rede. O aumento de dispositivos ligados à rede e a sua mobilidade tornam praticamente impossível a sua inventariação de uma forma manual, por parte do gestor da rede, daí a importância desta componente.

7. **Gestão dos recursos da rede:** As ferramentas de gestão de rede permitem ao administrador da rede gerir os diferentes dispositivos. Pode incluir várias funções tais como a obtenção do estado/eventos de cada dispositivo.

8. **Gestão do desempenho:** Conjunto de ferramentas que permitem ao gestor fiscalizar o desempenho e a disponibilidade da rede e dos seus diferentes componentes.

Existe uma grande apetência por parte dos administradores de sistemas de informação para a concentração deste conjunto de funcionalidades em uma só plataforma de gestão. Estas plataformas de gestão tomam a designação de plataformas de Gestão Empresarial (*Enterprise Management*). Actualmente existem no mercado três grandes fornecedores de plataformas de "Enterprise Management": Tivoli, Computer Associates e Hewlett-Packard.

Tivoli TME-10

Adquirida pela IBM, a plataforma Tivoli TME-10 [33] pertence à classe de produtos de gestão empresarial. É composta por um conjunto de funções integradas, construídas em redor de uma bancada de gestão denominada TME. A arquitectura do Tivoli assenta num *Object Request Broker* que se encontra conforme as especificações OMG/CORBA. A Tivoli tem como objectivo utilizar a bancada para esconder as diferenças entre os sistemas heterogénios por forma a gerir toda a variedade de sistemas, redes e aplicações.

O Tivoli oferece o seguinte conjunto de funcionalidades: Asset Management, Decision Suport, Inventory, Management Framework, Manager for Network Hardware, Problem Management, Remote Control, Security Management,Service Desk e Software Distribution. Os utilizadores do sistema Tivoli têm ainda a possibilidade de incorporar ferramentas de outros fabricantes.

O Tivoli suporta um leque muito variado de maquinaria e de sistemas operativos donde se inclui o OS/2 Warp, Windows 9x, Windows NT, AIX 4.2, Solaris 2.5 e HP/UX 10.x. Igual variedade encontra-se presente no suporte ao seu repositório de dados: DB2 2.11, Microsoft SQL Server 6.5, Informix 7.1, Oracle 7.2.x, Sybase System 10.

O Tivoli tem como principais limitações o facto de não incluir ferramentas de descoberta de recursos nem de gestão de licenças. A interface com o utilizador corresponde a uma aplicação que tem de ser instalado na estação cliente. Encontra-se em desenvolvimento uma interface baseada na WEB. Finalmente, os custos de instalação e manutenção do Tivoli são significativamente mais elevados que os apresentados pelo concorrentes.

Unicenter TNG

Produzido pela Computer Associates (CA)[4], o Unicenter TNG encontra-se organizado em torno de uma bancada de gestão. Contrariamente às restantes soluções de gestão empresarial, considera as diferentes componentes de gestão como módulos. Estes módulos são aplicações de gestão que podem ser ligadas à bancada de gestão. Desta forma, cada organização pode adquirir cada um dos módulos separadamente, adequando o seu parque de produtos de gestão às suas necessidades. Em complemento aos módulos de help-desk, controlo remoto, inventário, distribuição de software, gestão de problemas e de segurança que normalmente constituem as plataformas de gestão empresarial, a Computer Associates suporta um conjunto de ferramentas fornecidas por terceiras partes, donde se inclui a PowerSoft e a SAP, o suporte a bases de dados como o DB2 ou Microsoft SQL Server, suporte a sistemas operativos como VMS ou UNIX, etc.

Na arquitectura do Unicenter TNG, os gestores e agentes têm uma relação n-para-n, ou seja, um gestor pode gerir múltiplos agentes e cada agente pode ser gerido por múltiplos gestores. Esta abordagem possibilita a constituição de soluções de gestão tolerantes a faltas. Além disso gestores podem agir como agentes de outros gestores constituido assim um modelo hierárquico de gestão.

O Unicenter TNG suporta um conjunto vasto de maquinaria, sistemas operativos e bases de dados. Inclui facilidades tais como ferramentas de visualização (2D, 3D, e "grande-angular") e interfaces de visualização baseadas na Web. Dispõe de soluções vanguardistas tais como agentes neuronais (*neugents*), permitindo delegar nos agentes actividades de gestão que anteriormente eram da responsabilidade da estação de gestão de rede. Disponibiliza também um vasto conjunto de ferramentas de gestão, que cobrem o maior número de disciplinas da gestão dos sistemas de informação. Esta característica tem um preço: o Unicenter só é ultrapassado pelo Tivoli no que respeita a custos de instalação e manutenção.

HP OpenView

Desenvolvido pela Hewlett-Packard, o OpenView [52] disponibiliza funções de inventário de hardware e software, distribuição de software, gestão de segurança, gestão de armazenamento e gestão de desempenho. Adicionalmente, outros fornecedores disponibilizam opções para a gestão dos mais variados componentes, nomeadamente, gestão de bases de dados como o Informix, Oracle ou Sysbase, gestão de aplicações como o

SAP, PeopleSoft e Baan, gestão de sistemas tais como a maior parte dos sistemas UNIX, Microsoft Windows 9x, Microsoft Windows NT, Novell NetWare e OS/2. As grandes limitações do OpenView revelam-se ao nível das ferramentas de suporte à gestão de problemas: não incluí ferramentas de help-desk nem de controlo remoto. Nesta área a HP depende de produtos de terceiros (Remedy e Symantec PCAnywhere). Para além disso, o facto de o seu repositório de dados ser um ficheiro com um formato proprietário (flat file) limita a manipulação dos dados, nomeadamente na produção de relatórios.

Conceptualmente, estas plataformas apresentam, relativamente aos sistemas de gestão especializados, vantagens inerentes à integração de funções nomeadamente: a correlação de ocorrências e a visão global dos sistemas de informação. No entanto, a realidade tem demonstrado que a adopção destes sistemas se traduz em elevados custos e tempo na configuração e adaptação às necessidades dos utilizadores.

Sumário

Este capítulo fez uma panorâmica dos problemas abordados pelas plataformas de gestão de redes assim como das arquitecturas e mecanismos desenvolvidos para os resolver. Foi feita também uma breve descrição das principais plataformas de gestão de redes.

Verificou-se que grande parte dos trabalhos realizados nesta área se focam na interacção entre a estação de gestão e os elementos da rede. Esta materializa-se essencialmente em diferentes propostas de protocolos de gestão de rede e de definição das responsabilidades de gestão atribuidas a cada entidade envolvida.

Capítulo 3

Gestão de Vistas

3.1 Vistas na gestão de redes

A gestão eficiente de uma rede complexa e heterogénea requer a conciliação de dois requisitos aparentemente contraditórios: a gestão integrada de todos os elementos associados directa ou indirectamente ao sistema alvo e a gestão especializada de cada um desses elementos.

A componente integrada da gestão é importante para proporcionar uma visão global do sistema, condição essencial à correlação de eventos e ao consequente tratamento pró-activo dos problemas. Uma fonte de alimentação não-interrompível, por exemplo, embora não possa ser considerada um elemento da rede, pode ter uma importância fundamental para o seu bom funcionamento. A integração da sua gestão conjuntamente com a gestão dos restantes elementos da rede possibilita a obtenção de um conjunto de mais valias interessantes: maior exactidão na determinação de falhas, tratamento pró-activo dos problemas, etc.

A componente da gestão especializada está directamente relacionada com a complexidade da gestão das tecnologias da informação em geral e dos elementos da rede em particular. Esta complexidade obriga à especialização dos seus administradores na gestão de determinados componentes, sendo praticamente impossível um administrador de rede conseguir cobrir todas as diferentes especializações.

É na integração destas duas realidades que a noção de *vista* adquire um relevo significativo. Perante um sistema de gestão integrado, com a capacidade de gerir os múltiplos elementos de que se compõe a rede, as vistas disponibilizam representações mais reduzidas da rede, potenciando desta forma a gestão especializada, sem que isso impossibilite a visão universal da rede.

3.1.1 Definição de vistas

Retomemos o exemplo descrito na Secção 2.2. Devido à dimensão do banco, o seu administrador viu-se obrigado a distribuir as funções de gestão pelos gerentes dos balcões. No entanto, não foi privado da informação necessária à gestão global do banco: a informação que lhe chega vinda dos balcões é em menor quantidade mas de melhor qualidade, reflectindo o estado de cada balcão. A distribuição/delegação de funções de gestão não é sinónimo de perca de controlo por parte do administrador mas sim de uma gestão mais eficiente.

Algo de semelhante se passa nas soluções de gestão de rede. Embora as soluções, protocolos e arquitecturas de gestão de rede sejam tendencialmente distribuídas, continua-se a centralizar o controlo global da rede. Uma plataforma de gestão de rede suporta um conjunto variado de protocolos e arquitecturas que permitem distribuir as funções de gestão, quer através da existência de uma hierarquia de estações de gestão, quer pela incorporação dessas funções nos componentes geridos. Apesar disto, continuam a ser necessários mecanismos que permitem observar e controlar centralmente toda a rede, sem que para isso o administrador seja obrigado a actuar directamente sobre os diferentes componentes nos quais foram delegadas funções de gestão.

Resumindo, na gestão de rede podemos encontrar duas realidades ou competências distintas: a execução de funções de gestão de rede e o controlo global da rede. A execução de funções de gestão incide claramente sobre as propriedades das entidades geridas, sendo abordada de modo abrangente pelas diferentes soluções, protocolos e arquitecturas de gestão de rede existentes. O controlo global da rede proporciona uma visão global da rede e dos seus respectivos componentes e das diferentes perspectivas passíveis de serem dela extraídas. Pode-se considerar que a execução de funções de gestão de rede actua directamente sobre os diferentes elementos da rede enquanto o controlo de rede interage sobre a informação que caracteriza a rede, utilizando-a para apresentar as suas diferentes perspectivas. Esta informação encontra-se espelhada numa representação da rede.

Nas soluções centralizadas de gestão, estas duas competências encontram-se concentradas na estação de gestão de rede. Neste cenário, a fronteira entre a execução de funções de gestão e o controlo da rede é muito ténue, sendo por vezes difícil a sua distinção. A separação de competências manifesta-se mais claramente nas soluções distribuídas. A figura 3.1 descreve esta dualidade das plataformas de gestão de rede.

Tipicamente, nas plataformas de gestão de rede a representação da rede toma a forma de um conjunto de dados (bases de dados, ficheiros, etc.) que contém a informação necessária à inventariação e caracterização dos seus elementos. Uma forma de assegurar a integração da informação consiste em colocar todos os dados num único repositório que conterá toda a informação necessária à descrição e caracterização da rede e seus elementos, tornando possível a visualização e manipulação de toda a rede a partir uma única estação de gestão. Esta ideia, apesar de interessante, é cada vez menos exequível à medida que aumenta a complexidade e a heterogeneidade da rede.

A gestão eficiente de uma rede complexa e heterogénea obriga à especialização dos seus administradores na gestão dos diferentes elementos que nela se encontram inse-

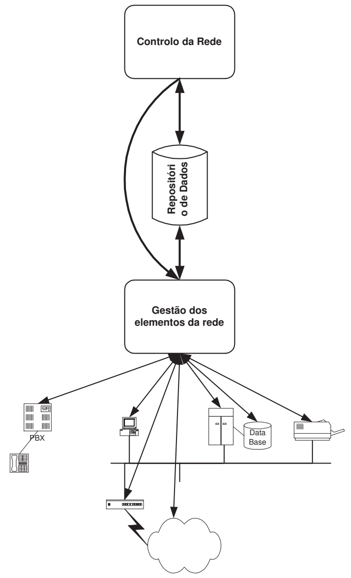

Figura 3.1: Componentes de controlo e gestão da rede

ridos. Numa rede complexa, não fará sentido o administrador dos equipamentos de encaminhamento gerir também os computadores pessoais, ou o administrador dos sistemas de bases de dados ter acesso à informação de gestão dos equipamentos de comutação. A existência de uma única representação da rede poderá ser um constrangimento à própria gestão, uma vez que se sobrecarrega os seus administradores com um excesso de informação perfeitamente irrelevante para as suas funções.

Perante este cenário colocam-se duas alternativas: *i)* separar a gestão da rede nas suas diferentes disciplinas, criando desta forma plataformas de gestão distintas e especializadas, ou *ii)* sobre uma mesma representação da rede apresentar as diferentes perspectivas da rede que contêm apenas a informação necessária a cada área da gestão.

A primeira alternativa, criação de plataformas especializadas de gestão, dificulta a integração da informação referida anteriormente, uma vez que passamos a ter várias fontes de informação de gestão desintegradas, o que dificulta a visão global da rede e a correlação da informação originária nos diferentes elementos da rede. A título de exemplo, considere-se a existência de plataformas distintas de gestão dos equipamentos de rede (por exemplo, *hubs*, encaminhadores) e de gestão dos sistemas (por exemplo, Unix, Windows NT). Quando se verifica uma falha no acesso a um servidor, o administrador do sistema terá alguma dificuldade em identificar se foi falha do servidor ou se foi uma falha do equipamento de rede que suporta a sua ligação. Além disso, poderá ser difícil estabelecer fronteiras precisas entre as diferentes competências da gestão, o que pode levar à existência de informação e mecanismos de gestão redundantes.

A segunda alternativa parte do principio que toda a informação de gestão se encontra centralizada, sendo disponibilizado a cada administrador especializado somente a informação que lhe é relevante. Desta forma, é possível a convivência numa mesma solução de gestão de uma representação global da rede e das diferentes representações parciais ou especializadas. Estas múltiplas representações de uma mesma realidade definem o conceito de *vistas*. As vistas permitem o suporte à representação das diferentes perspectivas que a rede e seus elementos podem tomar. Além de facilitarem a interacção com o sistema de gestão são também um importante mecanismo de suporte ao controlo de acesso à informação de gestão.

Para melhor compreendemos as vantagens e os problemas das vistas, consideremos como exemplo uma estrutura de ficheiros de um sistema Unix em que cada ficheiro é caracterizado por um nó-índice (*inode*) e representado em um ou mais directórios. Se o sistema de ficheiros for apresentado ao utilizador num ambiente gráfico, os ficheiros são representados por ícones e os directórios são representados por pastas.

Conside-se que este sistema de ficheiros é partilhado por um conjunto de utilizadores com diferentes necessidades de informação. Neste caso, poderá ser interessante agrupar os ficheiros de acordo com os respectivos utilizadores. Desta forma cada utilizador terá disponível uma pasta contendo os ficheiros por ele utilizados. É ainda possível classificar ou hierarquizar os ícones através da sua inclusão numa árvore de sub-pastas. Para os casos de ficheiros que são acedidos por mais do que um utilizador, existem duas soluções: a replicação dos ficheiros ou a representação do ficheiro por mais do que um icone (ou pela criação de *links*, na caso do sistemas de ficheiros Unix). Na primeira alternativa levanta-se o problema da coerência dos dados, enquanto na segunda alternativa temos o problema do controlo da concorrência no acesso aos ficheiros partilhados.

Com este sistema de ficheiros é possível definir, agrupar e hierarquizar a informação conforme o perfil e as necessidades do utilizador. Neste exemplo, a unidade de partilha e configuração é o ficheiro. Por vezes, pode ser necessário uma menor granularidade na especialização, limitando por exemplo os registos do ficheiro acessíveis a cada utilizador. Isto facilita a interação (ao esconder do utilizador a informação que ele não necessita) e aumenta a segurança do sistema (impedindo o acesso a informação não autorizada).

Outro aspecto interessante da interface com o sistema de ficheiros consiste na utilização das extensões dos ficheiros por parte de aplicações específicas: quando pretendemos abrir um documento de Microsoft Word, por omissão a janela de abertura apresenta-nos somente os documentos com a extensão ".doc". Ou seja, perante o utilizador do referido processador de texto, o sistema de ficheiros reduz-se a ficheiros com uma determinada extensão, constituíndo-se assim uma vista do sistema de ficheiros. Este sistema de vistas permite agrupar os seus elementos segundo um único critério: a extensão do ficheiro. Este critério de constituição de vistas apresenta também uma particularidade interessante a nível da componente de segurança: por omissão os ficheiros de sistema (ficheiros com extensões "sys", "dll", etc) não são apresentados ao utilizador.

Nesta analogia com um sistema de ficheiros, considerámos que um ficheiro corresponde à representação de um elemento da rede cujos atributos correspondem aos registos desse mesmo ficheiro e que as vistas correspondem aos directórios ou pastas que agrupam as representações dos ficheiros. As referências para os ficheiros são apresentadas ao utilizador inseridas em um ou mais directórios ou pastas sob a forma de uma entrada de directório ou ícone. O conjunto de ficheiros (e ícones associados) reflectem o comportamento do componente que representam. Este comportamento pode corresponder à alteração da informação constante no nó-indice, à alteração da sua cor, etc. Esta abordagem possibilita um conjunto de funcionalidades interessantes:

- Agrupamento dos componentes da rede conforme as suas características e contextos de classificação. Por exemplo, um encaminhador pode estar incluído na pasta dos encaminhadores, na pasta do edifício onde se encontra instalado e dentro desta na pasta do respectivo piso. Por sua vez este conjunto de pastas pode estar incluído noutras pastas que as agrupam pela entidade responsável pela gestão, por região ou país, etc.

- Concretização de mecanismos de segurança ao nível das pastas, permitindo restringir o acesso aos elementos referenciados na pasta. Esta definição de segurança pode ainda ser definida ao nível de cada ficheiro, independentemente das restrições de acesso efectuadas ao nível das pastas/directórios que o referenciam.

- A informação sobre o componente da rede encontra-se concentrada num só ponto (o ficheiro), eliminando-se desta forma os problemas de incoerência, característicos da redundância.

Apesar destas funcionalidades, esta solução simplificada apresenta alguns problemas. O sistema de ficheiros do nosso exemplo não suporta as seguintes funcionalidades:

- O acesso à informação do componente da rede é efectuado segundo o princípio de "tudo-ou-nada": ou se tem acesso à informação constante do ficheiro ou não se tem acesso. Quando o acesso é permitido, o acesso é total e independente do contexto da pasta. O resultado desta limitação seria, por exemplo, o gestor do fornecimento de energia de um edifício, ao qual foi disponibilizada uma pasta contendo os diferentes elementos da rede organizados por quadros eléctricos, ter acesso à informação sobre a configuração do equipamento ou dos respectivos utilizadores.

- O comportamento dos múltiplos ícones que representam um mesmo ficheiro, correspondem unicamente ao comportamento do que este representa. Mais uma vez, o comportamento do ícone não é sensível ao contexto da pasta. Vejamos o seguinte exemplo: o facto do espaço em disco de um servidor ser inferior a um limite predefinido tem pouca relevância para o gestor do fornecimento de energia (é irrelevante ou mesmo contraproducente que o ícone que se encontra na pasta de fornecimento de energia reflicta este tipo de ocorrência).

- O comportamento de um ícone não é afectado pelo comportamento dos restantes ícones da pasta em que está integrado. A falha de um componente da rede pode ser determinante para o funcionamento de outros componentes. O reflexo desta condição no contexto da vista pode auxiliar o gestor no tratamento pró-activo de problemas.

Deste modo, a utilização dos elementos básicos de um sistema de ficheiros como base para a representação da rede e seus componentes num sistema de gestão apresenta-se como uma uma abordagem interessante mas insuficiente. No nosso entender um sistema de vistas adequado para a gestão de rede deverá resolver as seguintes necessidades:

- Permitir o conhecimento da rede como um todo, por forma a proporcionar a visão de todos os elementos da rede independentemente da sua função.

- Permitir a representação das múltiplas relações entre os componentes da rede de tal forma que seja possível classificar as relações e os componentes da rede nela envolvidos.

- Permitir a manipulação dos atributos dos componentes da rede.

- Permitir que a criação de relações entre os componentes da rede seja suficientemente flexível para possibilitar uma navegação coerente pelas representações da rede, independentemente do contexto da vista.

- Permitir a concretização de mecanismos de segurança que sejam sensíveis ao contexto da vista por forma a dar aos seus utilizadores unicamente os recursos estritamente necessários.

3.1.2 Suporte à execução

A noção de vistas é indissociável dos aspectos dinâmicos e de distribuição que caracterizam o funcionamento dos elementos da rede e do sistema de gestão da rede. Como vimos anteriormente, existem duas classes de sistemas de gestão que não satisfazem totalmente os requisitosactuais da gestão dos sistemas: os sistemas verticais especializados na gestão especifica de elementos de rede(ou de determinado fabricante) e os sistemas de gestão universais que virtualmente permitem a gestão de toda a rede através de uma única representação da rede.

Uma abordagem possível consiste na centralização da visão global da rede e na distribuição da gestão especializada proporcionando, desta forma, a existência de várias consolas de gestão que, olhando para um mesmo universo de componentes da rede, apresentam representações distintas. Este conceito de distribuição pode levantar alguns problemas durante a execução da aplicação, nomeadamente na interacção entre as consolas ou vistas do sistema de gestão e os elementos da rede.

A concretização mais simplista de um sistema que suporta várias consolas, passa por obrigar a que cada representação interaja directamente sobre o respectivo componente da rede. Considerando a representação do sistema de gestão descrita no capitulo anterior, significa isto que a manipulação de cada ícone corresponde a acções sobre o respectivo elemento da rede. Cada ícone tem associado um conjunto de actividades de diálogo com o elemento da rede que representa, independente do conjunto de actividades de outro eventual ícone que represente o mesmo elemento.

Esta arquitectura é suportada por diversas plataformas de gestão. Tipicamente, é concretizada através da existência de um ficheiro que contém a descrição da vista, dos seus elementos e das suas relações. A distinção de vistas materializa-se em diferentes ficheiros de configuração que podem apresentar diferentes formas de como os componentes se relacionam na rede ou na organização. A manipulação de uma vista corresponde obviamente à sua interpretação por parte do programa de gestão de rede que normalmente disponibiliza uma representação gráfica da rede, a definição e execução de procedimentos periódicos e automáticos de fiscalização, consulta a pedido dos atributos dos elementos e a actuação directa sobre os atributos e configuração dos elementos. Por sua vez, a manipulação de mais do que uma vista corresponde à existência de igual número de instâncias activas do programa de gestão de rede.

Esta abordagem é muito frequente nos chamados editores de rede que permitem dispor ícones num espaço gráfico e interligá-los por forma a representar as relações entre os elementos. Estas aplicações têm normalmente predefinidos procedimentos automáticos de fiscalização que podem ser activados sobre componentes específicos. Todo este desenho pode ser salvaguardado em ficheiro e posteriormente recuperado para manipulação. As Figuras 3.2 e 3.3 apresentam, com exemplo deste tipo de aplicações, respectivamente o sistema de gestão SNMPc da Castle Rock Computing [22] e um exemplo de um ficheiro de definição da representação da rede.

Esta forma de concretização de vistas tem obviamente muitos inconvenientes. Vejamos os mais evidentes.

- A existência de várias instâncias do sistema de gestão a executarem a mesma ou diferentes vistas tem como consequência a ocorrência de actividades redundantes

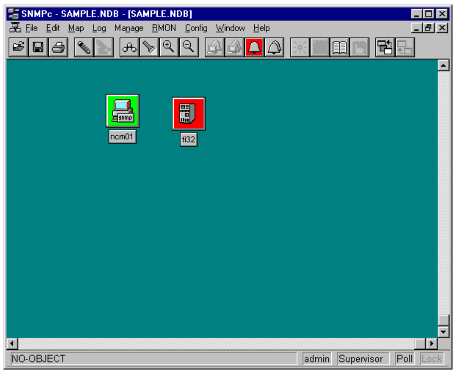

Figura 3.2: Exemplo de um editor de rede

```
VERS 2
PORTVARS  "sysObjectID.0" "sysServices.0" "*"
LFILT "                        246             26ac    0       0"
LFILT "                        4333            2648    0       0"
LFILT "                  Port $1 Up      4245          25e5    25e4    0"
LFILT "                  Port $1 Down    4223          25e4    25e5    0"

LFILT "                  Node Status Up  4245          2581    2580    0"
LFILT "                  Node Status Down        4222          2580    2581    0"
LFILT "                  Node responding*        4245          251d    251c    0"
LFILT "                  No response from node   4221          251c    251d    0"

NODE "fi32"
ADDR "129.101.27.5"
PORTVARS  "sysObjectID.0" "sysServices.0" "*"
ICON MINI.ICO
EXEC AUTO.EXE
POS -45 -116

NODE "ncm01"
ADDR "129.101.27.65"
PORTVARS  "sysObjectID.0" "sysServices.0" "*"
EXEC AUTO.EXE
POS -135 -120
```

Figura 3.3: Ficheiro de configuração da vista do sistema Castle Rock-SNMPc de fiscalização sobre os elementos da rede, que advém da fiscalização periódica definida em cada vista activa.

No desenho de soluções de gestão de rede, pretende-se um baixo consumo de recursos, sejam eles recursos computacionais necessários aos elementos da rede para correrem os respectivos agentes, ou recursos de comunicações. A existência de múltiplas vistas activas encontra-se longe de obedecer a este principio uma vez que se verifica uma utilização redundante e desnecessária quer dos recursos da rede quer dos recursos computacionais dos elementos da rede resultante da resposta a múltiplos pedidos idênticos originários das diferentes vistas.

Este problema é ainda mais evidente nas notificações geradas pelo elemento da rede uma vez que este terá de conhecer a existência de várias vistas activas para que possa, quando necessário, notificá-las de eventuais eventos. De notar que é comum que os equipamentos de rede só tenham a capacidade de notificar uma única instância do gestor.

- Esta classe de soluções de gestão de rede tem ainda o problema da gestão da coerência dos dados. Estando cada ícone associado a um conjunto de atributos que caracterizam o elemento da rede que representa, a existência de vários representantes do mesmo componente, referenciados nas diferentes vistas, tem como resultado uma proliferação de atributos redundantes que obrigam à criação de procedimentos de reconciliação da informação.

  Sendo cada vista um ficheiro, a solução deste problema é claramente mais organizacional do que técnica. A complexidade de concretização de uma aplicação de reconciliação dos dados motiva a criação de um sistema de vistas sem redundância de informação.

Este tipo de soluções, embora pareça desactualizado, continua presente em muitas

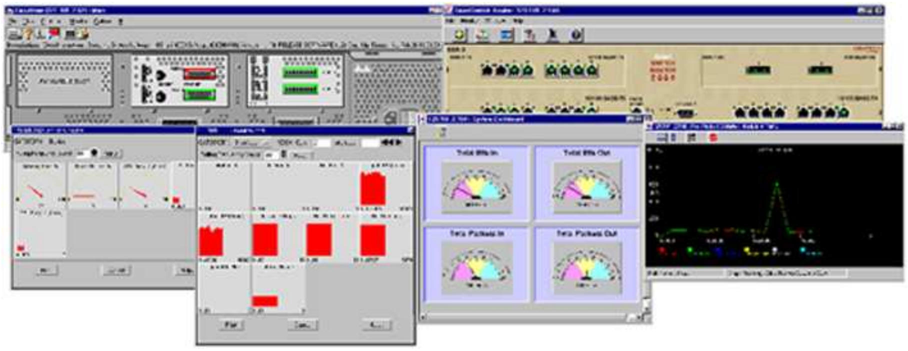

Figura 3.4: CiscoWorks (Cisco) e CoreWatch (Cabletron)

soluções de gestão actuais, nomeadamente em sistemas de gestão verticais especializados na gestão de elementos específicos da rede ou produtos de um determinado fabricante. Uma organização que tenha uma rede heterogénea e que suporte a sua gestão somente em sistemas de gestão especializados (gestores do encaminhador do fabricante x, gestor do comutador do fabricante y, gestor do sistema operativo z, etc), terá pela frente o trabalho algo inglório de manter a coerência das representações da rede em cada produto. Poderá também tentar criar mecanismos de reconciliação de dados entre plataformas, tarefa que na maior parte dos casos não é simplificada pelos fabricantes.

De notar, que não se pretende dizer com isto que as soluções especializadas não são importantes para as organizações. Tendo em conta o estado de arte da gestão de rede, os sistemas especializados são fundamentais para a gestão concreta dos componentes a que se destinam. Mas no nosso entender estes sistemas de gestão devem ser vistos não como o produto final da gestão, mas como ferramentas que devem ser integradas numa bancada que tenha a capacidade de proporcionar a visão global da rede e as múltiplas visões parciais ou refinadas com auxilio dessas ferramentas especializadas. A interface típica dos sistemas de gestão específicos é ilustrada na Figura 3.4.

As plataformas de gestão que assentam no modelo anteriormente referido, não endereçam a coexistência de uma visão global da rede e de múltiplas representações especializadas. Este tipo de soluções encontra-se distante do pretendido, tendo a sua apresentação nesta secção o objectivo de ilustrar algumas das dificuldades que se podem encontrar quando se pretende manter múltiplas representações de uma rede.

Na secção seguinte serão apresentadas algumas das plataformas de gestão que apresentam abordagens distintas na representação da rede e dos seus elementos.

3.2 Propostas de sistemas de vistas

Nos sistemas de gestão de rede vigentes distinguem-se basicamente dois tipos de soluções de representação da rede e seus elementos: soluções cuja unidade básica de representação é o componente da rede e soluções que têm como base os atributos desses mesmos componentes.

A primeira classe de soluções tem como objecto de tratamento os elementos da rede que se caracterizam por um conjunto de atributos. Podemos englobar nesta classificação desde soluções do domínio público a sistemas de "gestão empresarial". O Tkined [57], é um bom exemplo de soluções de domínio público. O CA Unicenter [4] e o HP OpenView [52] são exemplos de soluções comerciais cuja gestão assenta no elemento da rede. Engloba-se ainda nesta classe de soluções o sistema Domains.

A segunda classe de soluções refina a representação ao considerar os atributos dos elementos como unidade básica. Neste tipo de soluções, cada componente da rede é visto como o agrupamento de um determinado conjunto de atributos. É possível a criação de representações resultantes do agrupamento de atributos que não correspondem a nenhum dos elementos reais da rede. Se na primeira classe de soluções e no que diz respeito à semântica da representação da rede não se verifica grandes diferenças de conceito entre elas, o mesmo não se passa nesta segunda classe. Verificam-se aqui grandes diferenças de representação sendo possível encontrar desde representações de rede baseadas em folhas de cálculo a soluções que permitem interagir com a representação da rede através de SQL, passando por soluções cujas vistas são agrupamentos de atributos.

Iremos apresentar nos parágrafos seguintes uma breve resenha de como estas soluções representam a rede.

3.2.1 Domains

A necessidade de agrupar os objectos de gestão como forma de distinguir as múltiplas representações em que estes podem estar envolvidos, encontra-se espelhado no mecanismo de vistas proposto no sistema Domains [61, 42].

Neste sistema, os elementos alvo da gestão são representados por objectos que podem assumir múltiplas interfaces nas suas diversas facetas (interface com o utilizador, com os elementos alvo da gestão, etc). A multiplicidade de representações que objecto pode apresentar é materializada neste sistema através da sua associação a diferentes grupos de objectos ou *Domínios* (Domains). Os Domínios definem um conjunto de regras e restrições que permitem classificar e agrupar os objectos que cumprem esses requisitos, definindo desta forma os diferentes domínios da sua gestão.

A hierarquização dos objectos é conseguida através da possibilidade dos Domínios poderem ser membros de outros Domínios, tomando neste caso a designação de Sub-Domínios. A caracterização e validação dos Domínios, efectuada através do estabelecimento de um conjunto de regras e restrições (Por exemplo: "Os processadores deste domínio devem suportar o conjunto de instruções do M68000" ou "A remoção de um objecto deve apenas ser possível se este for membro de outro domínio") é concretizada pela verificação das regras aquando da criação e destruição dos objectos ou a sua inclusão ou remoção do Domínio. Claramente este sistema não prevê o carácter dinâmico que os objectos podem apresentar dado que as restrições não são validadas quando se modificam os atributos dos objectos.

Este sistema preocupa-se ainda com a caracterização dos utilizadores e respectivos acessos aos objectos. Os utilizadores são considerados objectos que também podem ser classificados e agrupados em domínios. O controlo de acesso é conseguido através do estabelecimento de regras de acesso entre domínios. Para tal, pode ser definido num Domínio uma lista de controlos de acesso (ACL, Access Control List) cujos elementos (ACE, Access Control Entries) descrevem o conjunto de operações e restrições que regulam o acesso a este de outros domínios.

Os conceitos em que assenta o sistema Domains encontram-se presentes no diferentes sistemas de gestão de rede actuais. O sistema Domains é, no nosso entender, uma referência para os sistema de vistas.

3.2.2 CA Unicenter

O CA Unicenter [4] representa a rede e seus componentes através um modelo orientado aos objectos. A figura 3.5 apresenta um exemplo da interface gráfica desta plataforma. Os componentes básicos de representação da rede são as classes e os objectos. As classes definem os diferentes tipos de elementos da rede, suas propriedades (atributos e relações) e comportamentos. Os objectos, instancias de uma determinada classe, representam um elemento concreto da rede.

As classes e os objectos deste sistema de gestão, são armazenados num repositório de objectos (Common Object Repository) suportado sobre uma base de dados relacional (Microsoft SQL server, Oracle, OpenIngres, etc). As classes encontram-se organizadas numa hierarquia, cuja expressão mais básica se encontra descrita na Figura 3.6.

Destacam-se duas classes: *Association Class* e *ManagedObject Class*. A Association Class define as relações possíveis entre os objectos. No sistema da CA, as relações entre os objectos encontram reduzidas a relações de inclusão e relações de ligação. As ligações podem ser explicitas (quando existe um objecto que a define) e implícitas (quando não

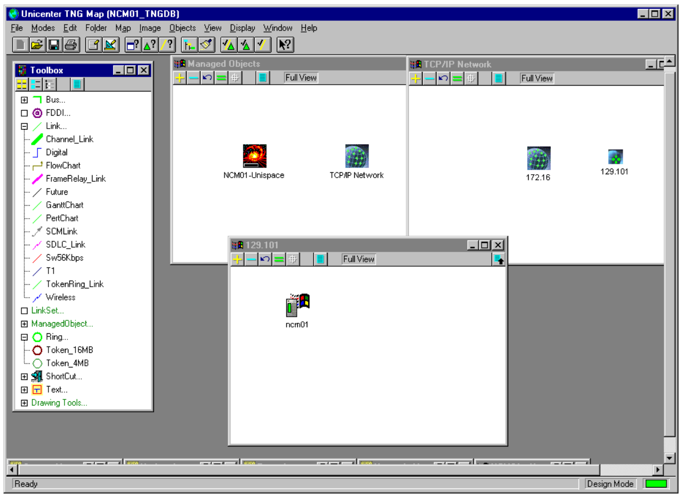

Figura 3.5: Interface grafica do CA Unicenter TNG

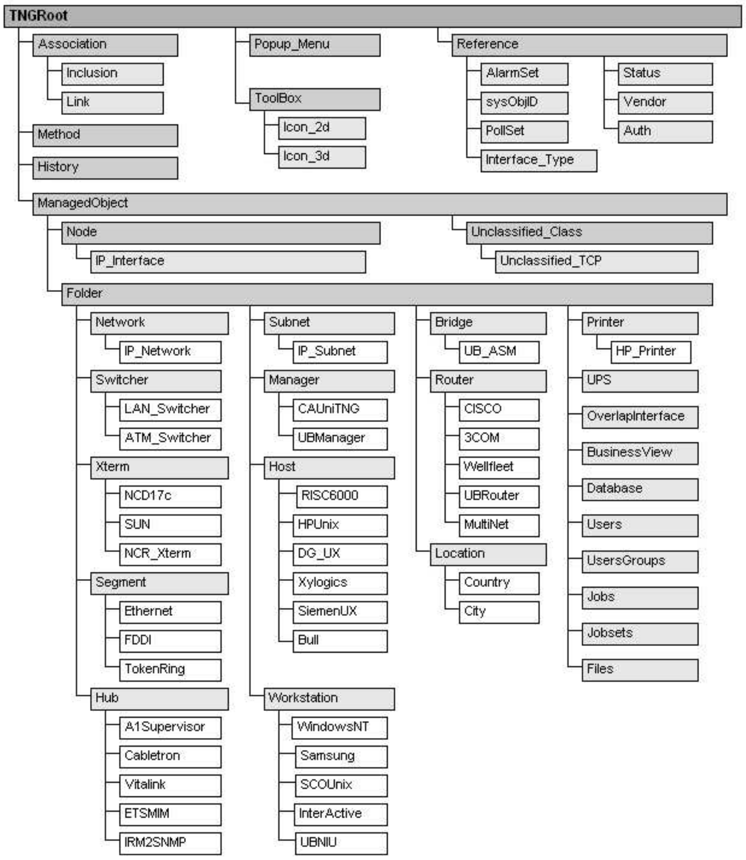

Figura 3.6: Hierarquia de classes do Unicenter TNG

existe nenhum objecto a defini-la como é o caso de dois PCs do mesmo segmento de rede).

A classe *ManagedObject* define os diferentes tipos de elementos que podem ser geridos pelo sistema. Destacam-se os *Nós* e as *Pastas*. Os nós definem elementos da rede que não contêm outros elementos; as pastas definem elementos da rede que podem conter outros elementos (relação de inclusão). De notar que um *ManagedObject* tanto pode ser a representação de um elemento concreto da rede, de um segmento de rede ou resultante do agrupamento de vários elementos concretos da rede.

No Unicenter TNG, a representação da rede é suportada por dois modelos: *modelo de objectos* e *modelo de topologia*. No modelo de objectos são definidas as classes e respectivas instâncias. Este modelo preocupa-se com a caracterização dos atributos e comportamento dos elementos da rede. O modelo topológico tem como função definir as relações entre objectos. O facto de ser separado do modelo de objectos permite que um objecto possa estar representado em múltiplas relações com outros objectos.

O sistema de gestão da CA, apresenta uma grande flexibilidade na representação da rede graças ao facto de ser baseado num modelo orientado por objectos. A caracterização do elemento da rede num esquema a três passos (definição da classe, instanciação da classe, definição do modelo topológico), permite por um lado reutilizar e refinar as definições dos elementos de rede e por outro separar claramente as componentes de caracterização dos elementos da rede das suas respectivas relações.

Apesar disso, a representação da rede no Unicenter apresenta algumas lacunas. Neste sistema, os atributos e o comportamento do objecto não são sensíveis ao contexto das vistas onde se encontra inserido. O Unicenter não suporta herança múltipla o que implica alguma redundância na definição das classes.

3.2.3 HP OpenView

A representação da rede, segundo a solução da HP [52], assenta em quatro conceitos: *objectos*, *mapas*, *sub-mapas* e *símbolos*. Os objectos representam internamente a entidade lógica ou fisica ou o recurso que existe algures na rede e que se pretende representar ou gerir através da solução de gestão de rede. Cada objecto caracteriza-se por um identificador e por um conjunto de campos que definem os atributos do objecto. Os mapas são uma colecção de objectos e respectivas relações. Tal como os objectos, os mapas não são directamente visíveis. Os símbolos são elementos de apresentação que representam um objecto num sub-mapa. Um objecto pode ser representado por múltiplos símbolos. Finalmente, um sub-mapa é uma colecção de símbolos que pretende representar uma determinada relação entre os objectos. Os sub-mapas e os símbolos são o suporte às vistas no sistema de gestão da HP. A hierarquização dos objectos é possível através da constituição de uma hierarquia de sub-mapas. Cada sub-mapa pode ser caracterizado pelo tipo de ligações que vai representar. Estes tipos encontram-se predefinidos e podem caracterizar desde ligações ponto-a-ponto, a redes com topologia bus ou anel, etc, como se ilustra na Figura 3.7.

A solução da HP permite a criação de vistas, possibilitando a existência de múltiplas representações das relações entre os objectos. A separação entre a componente de apresentação (símbolo) e componente de caracterização do elemento da rede (objecto) tem a vantagem de proporcionar a existência de várias vistas concorrentes sem ocorram incoerências nos dados. Isto acontece porque a caracterização do componente da rede encontra-se centralizada no objecto.

No entanto, a solução da HP apresenta como limitação a impossibilidade de adequar os símbolos ao contexto do sub-mapa: campos dos símbolos são os campos dos objectos, não sendo possível a um símbolo apresentar somente um subconjunto dos atributos adequados ao contexto da vista.

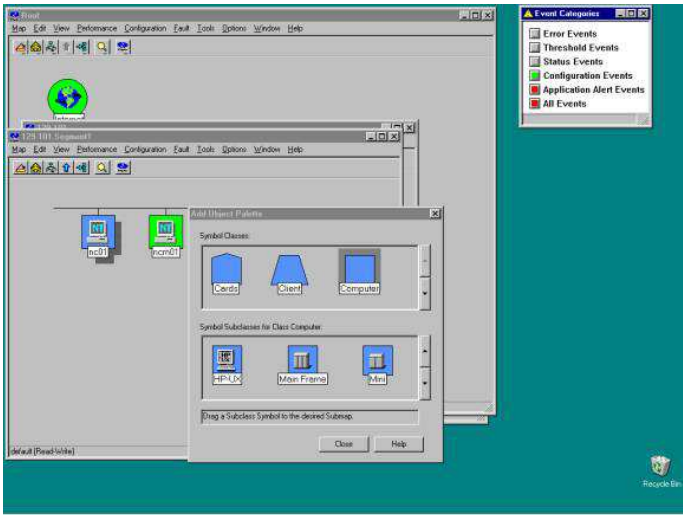

Figura 3.7: Exemplo da interface do Network Node Manager (OpenView)

A solução da HP, disponibiliza mecanismos de propagação de eventos na hierarquia de sub-mapas. Ou seja, quando ocorre um evento num objecto que, por exemplo, provoque a alteração do seu estado, verifica-se esta alteração nos respectivos símbolos e respectiva hierarquia de sub-mapas. Esta alteração de estado não é sensível ao contexto da vista, problema já referido anteriormente.

3.2.4 Tkined/Scotty

O sistema Tkined/Scotty disponibiliza dois mecanismos distintos para representar a rede: `ined` e o `map`. O `ined` permite definir a representação da rede e reflecti-la na componente gráfica do sistema (Tkined). O `map` tem um papel de repositório da informação e de suporte à caracterização do comportamento dos elementos da rede.

O Tkined/Scotty [57] assenta a sua representação da rede nos seguintes conceitos: *Rede* (Network), *Nó* (Node), *Elo* (Link) e *Grupo* (Group). A rede permite interligar um conjunto de Nós representando uma determinada topologia. O nó pretende representar os diferentes elementos da rede. O elo permite interligar nós a outros nós ou a redes. O grupo permite agregar e hierarquizar os elementos da rede. Ao `map` foi acrescentado um quinto elemento: *Porta* (Port) que permite definir as interfaces de um nó.

As características de um elemento da rede podem ser espelhadas nesta representação através da definição de atributos. Cada objecto (Rede, Nó, Elo ou Grupo) tem um conjunto de atributos pré-definidos (identificador, nome, endereço) aos quais podem ser acrescentados atributos definidos pelo utilizador. Por exemplo:

```
# Criação de um nó
set node01 [ined create NODE]
# Definição de atributos pré-definidos
ined name $node01 "ttl1"
ined address $node01 "192.168.10.1"
# Definição de atributos especificos
ined attribute $node01 localizacao "Lisboa"
ined attribute $node01 contacto "Sr. Silva"
```

O `map` além de definir os elementos da rede disponibiliza um mecanismo de caracterização dos respectivos comportamentos através do tratamento de eventos. O `map` será alvo de uma análise pormenorizada mais adiante.

Como se pode constatar, a visão da representação da rede para o Tkined/Scotty é bastante limitada, restringindo-se à apresentação dos componentes da rede e suas interligações. Pode-se afirmar que este sistema não dispõe de um mecanismo de vistas uma vez que diferentes representações do mesmo objecto só podem ser concretizadas através da criação de objectos distintos.

3.2.5 Marvel

Nos sistemas que descrevemos anteriormente, uma vista encontrava-se associada a representações particulares ou refinadas das relações entre os elementos da rede.No sistema Marvel [3, 2], o conceito de vista encontra-se associado à representação de uma agregação de atributos, podendo estes estar directamente relacionados com proprieda-

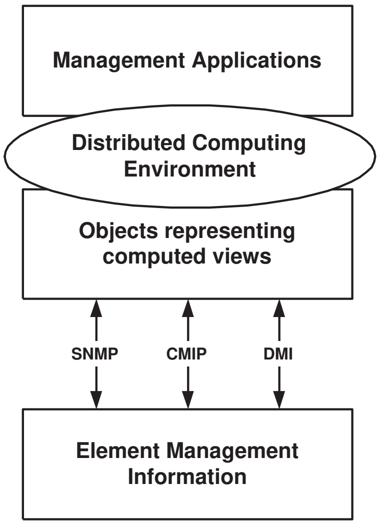

Figura 3.8: Modelo de informação do Marvel

des dos elementos da rede ou serem o resultado a computação de vários atributos de um mesmo elemento da rede ou de múltiplos elementos.

O objectivo do sistema Marvel é permitir a abstracção dos protocolos de gestão de rede, através da criação de um nível aplicacional intermédia que se encarrega de interagir com estes. Neste sistema, as vistas são construídas através de um processo de agregação da informação aplicado à informação de gestão recolhida dos elementos da rede. Cada vista é armazenada num objecto e pode agregar os seguintes componentes:

- Componente de fiscalização que contém a informação recolhida dos elementos da rede.

- Componente de controlo que representa uma interface para controlo das aplicações de gestão

- Componente de eventos que representa as notificações que foram geradas pelos elementos da rede.

Quando uma vista é criada, o administrador da rede define o conjunto de regras de agregação que caracterizam cada atributo do respectivo objecto. A agregação pode ser especificada pelo administrador da rede, através de uma linguagem estruturada ou pela associação de um pedaço de código ao atributo. As agregações de atributos podem ser espaciais ou temporais.

As agregações espaciais pretendem reduzir a informação de gestão através da sua catalogação em grupos coerentes. Esta agregação pode ser efectuada pelo administrador da rede baseada em qualquer critério coerente. O critério da localização do elemento da rede ou das suas funcionalidades são bons exemplos deste tipo de agregações.

As agregações temporais possibilitam o tratamento da informação que se encontra sujeita a variação com o tempo. Para tal, o sistema recolhe periodicamente informação da rede que servirá de base à função de agregação. Esta particularidade do sistema pode ser utilizada na correlação de informação ou na constituição de séries com diferentes granularidades temporais.

Este sistema tem a virtude de considerar que um componente de gestão rede não é necessariamente o mesmo que um elemento da rede. A representação de um elemento da rede na vista é uma das possíveis representações. Isto torna o sistema de informação do Marvel, mais abrangente que os sistemas de gestão que vimos anteriormente.

3.2.6 Folha de cálculo

A utilização do paradigma da folha de cálculo na representação e gestão da rede foi proposto por P. Kalyanasundaram et *Al* [35]. O paradigma da folha de cálculo assenta na abstracção das células: uma folha de cálculo é composta por células distribuídas numa matriz bidimensional referenciada por linhas e colunas. Uma célula é composta por duas componentes: informação de controlo e dados. A informação de controlo contém as regras de aquisição e manipulação dos dados. A componente de dados contém a informação resultante da concretização das regras.

Nesta proposta a informação de controlo é descrita através da linguagem SSL (Spre-

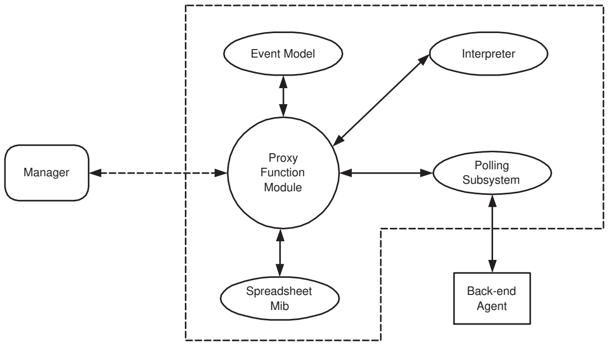

Figura 3.9: Arquitectura do sistema de gestão baseado do paradigma da folha de cálculo

adsheet Language). A Figura 3.9 descreve a arquitectura da solução.

Nesta solução o módulo *spreadsheet MIB* concretiza a abstracção da folha de cálculo através da manutenção de duas estruturas de dados: *tabela de controlo* (Control Table) e *tabela de dados* (Data Table). Na óptica do gestor da rede (Manager) esta componente é vista como uma MIB (ssmib) que reflecte o conteúdo destas duas tabelas.

A tabela de controlo, composta por uma sequência de registos (*controlEntry*), é responsável pela manutenção da informação de controlo das células. Cada registo contém um conjunto de elementos identificativos da folha a que pertence, dono, posicionamento na folha de cálculo e regras de controlo materializadas na linguagem SSL. Na Figura 3.10 é apresentado um exemplo da utilização da linguagem SSL. A tabela de dados, composta por uma sequência de linhas designadas por *dataEntry*, contém o resultado da execução das regras definidas na tabela de controlo.

O módulo *representante* (proxy), aceita pedidos de SNMP, utilizando o módulo *ssmib* para actualizar a informação ou para obter respostas. Caso existam regras de actualização periódica da informação (polling), estes pedidos são executados de uma forma optimizada pelo módulo de polling que interage com os agentes através de SNMP.

O módulo *modelo de eventos* (Event Model), é responsável pela gestão de eventos. Neste paradigma, um evento pode sinalizar a alteração de uma célula (controlo ou dados), uma alteração do sistema (recepção de uma mensagem SNMP, incremento do re-

```
Cell [1,1]:
        label: cellMibViewExample
        action:
        {
                // Poll counters at sol
                $$.1 = tcpActiveOpens@sol.cis.udel.edu
                $$.2 = tcpPassiveOpens@sol.cis.udel.edu
                $$.3 = ipInReceives@sol.udel.edu
                $$.4 = ipInHdrErrors@sol.udel.edu
                // Poll counters at tweety
                $$.5 = tcpActiveOpens@tweety.cis.udel.edu
                $$.6 = tcpPassiveOpens@tweety.cis.udel.edu
                $$.7 = ipInReceives@tweety.udel.edu
                $$.8 = ipInHdrErrors@tweety.udel.edu
                // Summing
                $$.9  = $$.1 + $$.2 // Total TCP connections at sol
                $$.10 = $$.5 + $$.6 // Total TCP connections at tweety
                $$.11 = $$.3 + $$.4 // Total IP datagrams receided with
                                    // no header errors at sol
        }
```

Figura 3.10: Exemplo de SSL

lógio, etc.) ou a execução de uma ou mais células da folha. São suportados eventos básicos (timer - gerado em cada incremento do relógio, pool - gerado quando ocorre a recepção de uma resposta a um pedido periódico, alteração do conteúdo de uma célula, etc.) e eventos definidos pelo utilizador. O mecanismo de eventos deste sistema proporciona uma flexibilidade muito grande na definição de comportamentos associados aos componentes da rede.

A utilização do paradigma da folha de cálculo para simplificação da gestão de rede parece ser uma proposta interessante quando se pretende tratar e correlacionar informação originária de vários componentes. O facto de suportar o tratamento de eventos proporciona ainda a constituição de mecanismos pró-activos e reactivos na detecção e recuperação de falhas da rede. No entanto, sendo uma solução vocacionada para o tratamento de informação que emerge dos elementos da rede, dificilmente consegue reflectir diferentes representações tendo em conta a relação entre elementos. As questões levantadas no capítulo 3.1 relativas à necessidade da representação dos elementos ser sensível ao contexto da vista são dificilmente endereçadas com esta solução.

3.2.7 SNMPQL

O SNMPQL, [73], propõe a utilização das ferramentas e interfaces tipicamente associadas à manipulação de bases de dados relacionais, no tratamento da informação de gestão de rede. O SNMPQL pretende manipular os objectos de SNMP através de uma sintaxe de SQL. Este modelo tem como premissa considerar-se uma MIB como uma base de dados relacional, ou seja, um conjunto de tabelas que por sua vez são constituídas por atributos.

Como vimos anteriormente (Secção 2.3.1), uma MIB é uma colecção de objectos, cuja instanciação corresponde à caracterização de um elemento concreto da rede. No SNMPQL, os objectos correspondem às colunas de uma tabela e as instâncias às linhas da mesma tabela. Consideremos como exemplo o objecto `ipRouteMetric1` e um encaminhamento para a rede `10.0.0.0`. No SNMP, o `ipRouteMetric1` é um objecto e o `ipRouteMetric1.10.0.0.0` é uma instância do referido objecto. No SNMPQL, o `ipRouteMetric1` é uma coluna da tabela que contém uma linha por cada encaminhamento. Ou seja, ao encaminhamento para a rede `10.0.0.0` corresponderá uma linha na tabela, cuja coluna `ipRouteMetric1` é a métrica desse encaminhamento.

Esta proposta tomou a forma de um produto comercial [23] que disponibiliza uma interface ODBC (Open Database Conectivity) destinada à conversão de comandos SQL em pedidos SNMP 3.11. A adopção desta interface possibilita a utilização desta funcionalidade nas mais variadas plataformas de desenvolvimento, nomeadamente, Microsoft Access, Crystal Reports, Visual Basic, Microsoft Internet Information Server, Visual C++, etc. Na Figura 3.12 são apresentados alguns exemplos da utilização da linguagem SQL no mundo SNMP.

Esta solução, embora não esteja directamente relacionada com a representação da rede, permite a construção de diferentes vistas através da manipulação da informação constante dos elementos da rede. O snmpQL, tem vantagem de tornar os objectos

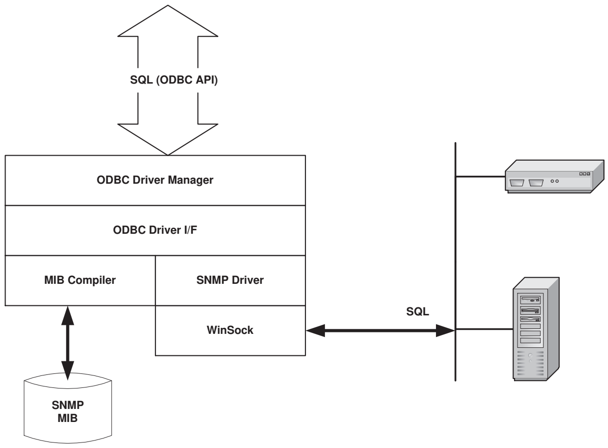

Figura 3.11: Arquitectura do SnmpQL

```
SELECT ipRouteDest, MIN(ipRouteMetric1) FROM ipRouteEntry GROUP BY ipRouteDest

SELECT system.sysDescr, system.sysUpTime FROM rmonprobe@system system;

SELECT system.sysDescr, system.sysUpTime FROM rmonprobe:161@system system;

SELECT sysDescr, sysUpTime FROM rmonprobe:161:public@system system;

UPDATE system SET sysDescr = 'xyz', sysObjectID = '1.2.3.4.5.6'

UPDATE udpEntry SET udpLocalPort = 1065 WHERE udpLocalPort = 1066
```

Figura 3.12: Exemplos de SnmpQL

SNMP acessíveis através de modelos de gestão de bases de dados.

Sumário

O Capítulo 2 deu-nos uma perspectiva da gestão de rede virada para a interacção entre os sistemas de gestão de rede e os elementos da rede. Este capitulo, focouum aspecto particular dos sistemas de gestão de rede: a sua representação. Foram assim analisadas as necessidades e problemas que advêm da representação da rede. Introduzimos o conceito de Vistas, ou seja, uma representação particular da rede com o objectivo de apresentar uma determinada perspectiva da relação entre os elementos da rede. Finalmente apresentou-se uma pequena panorâmica sobre as diferentes propostas de representação da rede e seus elementos.

Capítulo 4

Modelo de suporte à gestão de vistas

Este capítulo propõe um modelo para concretizar a noção de vistas. O modelo pretende ser suficientemente versátil para permitir a representação de realidades de gestão complexas e, ao mesmo tempo, suportar a observação coerente dos objectos geridos mesmo quando se executam concorrentemente actividades de gestão sobre vistas distintas.

Como iremos ver, este modelo sintetiza e integra diversos conceitos referidos anteriormente, os quais por vezes não se encontram suportados em plataformas especifícas de gestão de redes e cujas facetas se encontram actualmente concretizadas de modo disperso em sistemas independentes.

4.1 Modelo de objectos adoptado

4.1.1 Entidades elementares

O modelo considera as seguintes entidades: DISPOSITIVOS, ELOS, EMBAIXADORES, PASTAS, REPOSITÓRIOS as quais são introduzidas de seguida.

Dispositivo: representa um determinado objecto de gestão, tipicamente associado a um dispositivo físico ou lógico, como uma máquina ou um processo. Cada elemento alvo da gestão é representado por um único DISPOSITIVO que incorpora os seus atributos e comportamento. O dispositivo é também conhecido no nosso modelo por ELEMENTO NUCLEAR.

Embaixador: representa um dispositivo numa determinada perspectiva do universo de gestão, reflectindo total ou parcialmente os atributos do seu DISPOSITIVO. Um DISPOSITIVO pode ser representado por múltiplos EMBAIXADORES pertencentes ou não a múltiplas perspectivas do universo de gestão (VISTAS). Este componente é também conhecido por ELEMENTO.

Elo: representa uma ligação física ou lógica entre dois ou mais dispositivos numa determinada perspectiva do universo de gestão. Um ELO interliga EMBAIXADORES representando desta forma uma das múltiplas relações possíveis entre os DISPOSITIVOS.

Pasta: agregado de EMBAIXADORES e de PASTAS que permite estruturar os objectos de gestão de acordo com um critério organizacional. Uma PASTA pode assumir uma dualidade de papeis: entidade agregadora de entidades ou entidade representante de dispositivos complexos.

Vista: conjunto coerente de PASTAS, EMBAIXADORES e ELOS que pretendem representar uma das múltiplas perspectivas do universo de elementos da rede que se pretende gerir. Uma VISTA tem associadas RESTRIÇÕES que permitem concretizar políticas de controlo no acesso aos atributos dos dispositivos, como forma de aproximar o comportamento dos seus embaixadores ao contexto da vista. Esta entidade pode ainda ter associados filtros que definem as propriedades que os dispositivos deverão ter para que os seus embaixadores possam estar associados às VISTAS.

Repositório: É o componente do sistema que armazena os restantes elementos que vimos anteriormente. Um REPOSITÓRIO contém todos os representantes dos elementos da rede (DISPOSITIVOS) e todas as representações ou perspectivas em que esses elementos podem estar envolvidos (EMBAIXADORES, ELOS, PASTAS e VISTAS).

4.1.2 Atributos

Todas as entidades elementares possuem zero ou mais ATRIBUTOS. No nosso modelo distinguimos dois tipos de atributos, a saber:

Atributos simples: variáveis de tipo elementar que representam propriedades das entidades, como por exemplo inteiros ou cadeias de carácteres. Estes atributos podem ser utilizados para associar informação administrativa às entidades, para conter informação necessária ao acesso ao elemento da rede, para especificar parâmetros que controlam funções de fiscalização ou de teste do elemento, para armazenar informação provisória relativa aos processos de fiscalização, etc.

Métodos: Atributos associados a programas/funções. O valor de um atributo corresponde ao resultado da execução do respectivo programa. Estes programas podem ser executados a pedido (quando é solicitado o valor do atributo) ou com uma periodicidade pré-definida. Estes métodos podem ser utilizados para a disponibilização de atributos complexos que podem resultar da computação de atributos simples ou de atributos externos à entidade. Podem ainda ser utilizados para alterar, de uma forma periódica ou a pedido, os atributos das entidades do repositório ou mesmo actuar sobre os elementos da rede.

4.1.3 Restrições

Por defeito, um embaixador apresenta todos os atributos que caracterizam o dispositivo a que se encontra associado. As RESTRIÇÕES têm o objectivo de adequar os atributos dos embaixadores ao contexto da vista. Uma restrição é materializada numa lista de acesso associada à VISTA, que limita quais os atributos dos embaixadores que são observáveis na vista.

4.1.4 Filtros

Os FILTROS são regras que definem os critérios de inclusão de embaixadores numa vista. Um FILTRO permite a criação implicita numa vista, dos embaixadores que representam todos os dispositivos que cumprem as regras nele definidas. As regras dos filtros encontram-se associadas a valores que os atributos dos dispositivos devem conter.

Vimos anteriormente que um embaixador é um representante de um dispositivo numa vista. A criação/remoção de embaixadores numa vista pode ser efectuada de forma explicita ou implícita.

A criação explicita de embaixadores corresponde à criação dos embaixadores, no contexto de uma vista, associando-os aos respectivos dispositivos. Ou seja, um dispositivo só se encontra representado numa vista quando for expressamente determinado. A criação implícita de embaixadores é efectuada através da definição na vista de um conjunto de regras (filtro) que estabelecem os valores que determinados atributos dos dispositivos devem conter para que estes possam ser representados na vista. Quando os atributos de um dispositivo cumprem a especificação do filtro, é criado na respectiva vista um embaixador. De igual modo, o embaixador é removido de uma vista quando os atributos do dispositivo que representam deixarem de cumprir as especificações do filtro.

4.1.5 Associações

As entidades descritas anteriormente relacionam-se do seguinte modo:

Um DISPOSITIVO possui um ou mais ATRIBUTOS (simples e métodos) e referências para todos os seus EMBAIXADORES. O DISPOSITIVO possui a informação necessária para propagar eventos gerados no contexto de uma VISTA, para as restantes VISTAS em que está representado.

Um EMBAIXADOR possui uma referência para o DISPOSITIVO que representa, uma referência para a PASTA em que está inserido e referências para os ELOS a que está associado. Os ATRIBUTOS de um EMBAIXADOR são referências para os atributos do seu DISPOSITIVO.

Um ELO possui um ou mais atributos e referencias para os EMBAIXADORES a que está associado.

Uma PASTA possui um ou mais atributos, a referência para a PASTA em que está inserida (ou para a VISTA, caso se trate de uma pasta raíz) e uma lista de referências para as PASTAS, ELOS e EMBAIXADORES que contém.

Finalmente, uma VISTA possui um ou mais ATRIBUTOS, uma ou mais PASTAS, EMBAIXADORES ou ELOS e um conjunto de RESTRIÇÕES que definem quais os atributos dos embaixadores que podem ser visíveis nessa vista.

O modelo de objectos encontra-se ilustrado na Figura 4.1.

4.2 Modelo de eventos

O comportamento das entidades é gerido pelo processamento de eventos. O nosso modelo possibilita a tipificação dos eventos e a respectiva associação de procedimentos ao

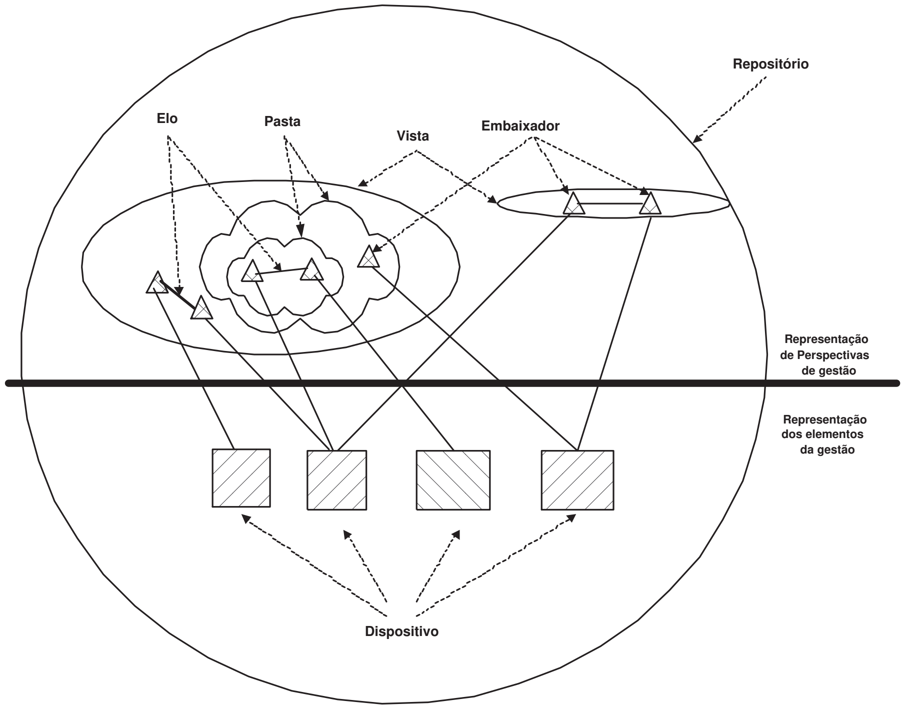

Figura 4.1: Modelo de objectos proposto

nível de cada entidade (dispositivo, pasta, embaixador, etc.), permitindo-se desta forma caracterizar o comportamento de cada entidade perante a ocorrência de eventos. Os eventos podem tipificar-se em duas classes distintas: *eventos internos* e *eventos externos*.

Os eventos internos reflectem acontecimentos associados à manipulação da ferramenta, como seja, alteração de atributos, adição e remoção de entidades a uma vista, etc. O tratamento a este tipo de eventos só se pode verificar no contexto da entidade onde o evento ocorreu.

Os eventos externos reflectem acontecimentos associados ao universo alvo da gestão. Dividem-se em três subclasses: eventos *ascendentes*, eventos *descendentes*, eventos *laterais*. O tratamento dos eventos externos pode propagar-se a mais do que uma entidade, sendo a forma de propagação dependente da subclasse de eventos.

- Caso ocorra um evento ascendente numa determinada entidade, o seu tratamento inicia-se nessa mesma entidade propagando-se pelas hierarquias superiores da

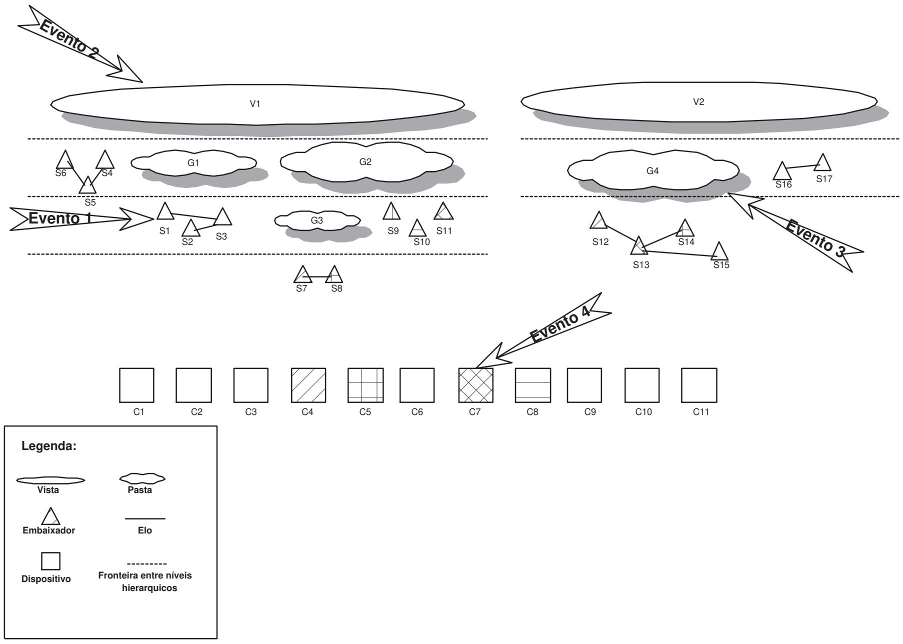

Figura 4.2: Exemplo da propagação de eventos

representação (pastas que contêm directa ou indirectamente o elemento onde ocorreu o evento).

- No caso de ocorrência de um evento descendente, o seu tratamento inicia-se na entidade visada e propaga-se por todas as hierarquias inferiores, podendo terminar ao nível do dispositivo.

- Finalmente, a ocorrência de uma evento lateral o seu tratamento é propagado através da malha definida pelos elos a partir da entidade origem. Esta propagação é limitada por um número de saltos (*hops*) pré-definido.

A Figura 4.2 permite ilustrar o funcionamento deste mecanismo de eventos. A ocorrência do Evento 1 no embaixador S1 poderá gerar a três comportamentos distintos:

- Se o Evento 1 for ascendente poderá ser tratado no embaixador S1, propagado e eventualmente tratado na pasta G1 e posteriormente propagado e eventualmente tratado na vista V1. O facto de o evento ser tratado ou não em cada uma das entidades por onde é propagado estará directamente relacionado com a necessidade de refinar o tratamento de determinada ocorrência. É assim possível, o tratamento genérico de uma determinado evento ao nível de uma vista ou de uma pasta ou um tratamento mais especializado ao nível do embaixador ou do dispositivo.

- Se o Evento 1 for descendente poderá ser tratado no embaixador S1 e posteriormente tratado no dispositivo C3. A ocorrência de eventos ao nível dos dispositivos tem a particularidade de gerar um evento ascendente que se propaga por todos os embaixadores do dispositivo. Caso o evento tenha origem num embaixador, este último não é alvo do evento ascendente gerado pelo seu dispositivo.

- Se considerarmos que o Evento 1 é lateral com um salto, será propagado para o embaixador S3.

A ocorrência do Evento 2 ao nível da vista V1 só fará sentido se estivermos na presença de um evento descendente. Neste caso, o evento será propagado e eventualmente tratado por todas as entidades pertencentes à vista V1. Este evento será ainda propagado para todos os dispositivos relacionados com os embaixadores da vista V1. Por sua vez, os dispositivos afectados pelo evento 2 darão origem a um evento ascendente que no caso concreto do exemplo apresentado, irão afectar as entidades da vista V2 (embaixadores, pastas e a vista V2).

De notar que cada entidade que é "visitada" pela propagação de um evento tem a capacidade de, no seu tratamento, interromper a propagação. Ou seja, o tratamento do Evento 1 pode circunscrever-se ao Embaixador 1; a propagação do Evento 2 pode ser interrompida ao nível do dispositivo não se verificando a ocorrência do evento ascendente ao nível das entidades da vista V2; etc.

Vejamos alguns casos concretos da aplicabilidade do mecanismo de vistas proposto:

Caso 1: Alteração do estado de um determinado atributo em resposta a um evento específico num dispositivo. A ocorrência do Evento 4 fig:eventos tem como consequência a sua propagação ascendente pelos embaixadores do dispositivo C7 (S11 e S13), pelos grupos G2 e G4 e pelas vistas V1 e V2. A alteração do estado de atributo anteriormente referido poderá ocorrer:

1. Pela execução de procedimentos especializados associados a cada uma das entidades envolvidas (S11, S13, G2, G4, V1 e V2).
2. Pela execução de procedimentos mais genéricos que se encontram associados ao nível das pastas ou das vistas. Desta forma, o tratamento de determinados eventos ao nível dos embaixadores poderá corresponder à simples associação dos mesmos a uma vista ou pasta que tenha associado o procedimento de tratamento desses eventos.

Este caso exemplifica a reflexo do estado de um elemento da rede (representado por um dispositivo) nas suas diferentes representações. De notar que a alteração do estado de um elemento da rede poderá não corresponder ao seu reflexo em todas as suas representações (embaixadores, pasta ou vistas). O mecanismo de restrições pode ser utilizado para limitar a visibilidade de atributos e eventos, permitindo adequar a representação dos elementos da rede ao contexto da vista.

Caso 2: Alteração do estado de um atributo em todos os elementos de uma pasta. Este caso poderá ser resolvido de duas formas:

1. Criação de um procedimento genérico ao nível da pasta que actua sobre todos os seus embaixadores.
2. Criação de procedimentos especializados associados a um determinado evento, ao nível dos embaixadores, resumindo-se a alteração do estado do atribu-

to à ocorrência de um evento descendente ao nível da pasta e sua respectiva propagação e tratamento ao nível dos embaixadores.

Os mecanismos de propagação de eventos propostos permitem:

- Reflectir, de uma forma selectiva, o estado dos elementos da rede nas suas diferentes representações (Eventos ascendentes)
- Possibilitar o tratamento de agrupamentos de entidades sem que seja necessário um conhecimento exaustivo das mesmas (Eventos descendentes).
- Reflectir e tratar a alteração de estado dos elementos da rede que se encontram relacionados entre si (eventos laterais).

Sumário

Neste capítulo foi proposto um modelo de suporte a vistas em sistemas de gestão de rede. O modelo distingue a caracterização dos elementos da rede das multiplas relações em que estes podem estar envolvidos.

O modelo prevê também a existência de um conjunto de meios que permitem caracterizar o comportamento da rede destacando-se os mecanismos de associação de metodos às diferentes entidades intervenientes no modelo e à propagação e tratamento de eventos dentro de uma mesma vista e entre vistas.

Capítulo 5

Concretização

De modo a validar a arquitectura proposta, foi desenvolvido um protótipo, materializado numa extensão Tcl passível de ser integrado na plataforma Tnm/Scotty. Como iremos ver, optou-se nesta concretização pela expansão de uma extensão existente (`Tnm::map`) em vez do desenvolvimento de raiz de uma nova extensão Tcl.

Esta secção apresentará uma descrição do Tcl e do Scotty, seguida da apresentação da extensão `Tnm::map`. Finalmente será apresentada a extensão que concretiza o modelo de vista anteriormente descrito.

5.1 Suporte à concretização

5.1.1 TCL: Tool Command Language

O Tcl é uma linguagem interpretada desenvolvida no inicio dos anos 80 por John Ousterhout [51]. Um programa em Tcl corresponde a um ou mais ficheiros contendo comandos textuais que são executados por um interpretador de Tcl. Existem dois interpretadores básicos: `tclsh` e `wish`. O `tclsh` é um interpretador TCL em modo consola que executa comandos `tcl` introduzidos de uma forma interactiva na sua consola ou

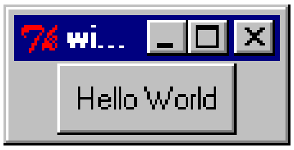

Figura 5.1: Botão "Hello World"

originários de um ou mais ficheiros de comandos. O `wish` é um interpretador gráfico que além de interpretar comandos Tcl, executa ainda comandos Tk. O Tk é uma extensão ao Tcl que suporta a criação e controlo de objectos gráficos. Vejamos como exemplo do Tcl e do Tk como poderá ser apresentada a cadeia de carácteres "Hello World".

Em Tcl, o comando `puts` pode ser utilizado para escrever no dispositivo de saída a frase pretendida:

```
puts "Hello World"
-> Hello World
```

Em Tk, a mesma frase poderá ser apresentada num botão:

```
button .hello -text "Hello Word"
pack .hello
```

A adopção do Tcl/Tk para o desenvolvimento de aplicações que suportam as mais variadas actividades, nomeadamente a gestão de rede, advém de algumas vantagens que a linguagem confere. A natureza interactiva intrínseca do seu interpretador, aliada ao facto dos programas em Tcl necessitarem de muito menos linhas de código que programas análogos feitos em C ou C++, proporciona uma forma rápida e simples de desenvolver aplicações. Além disso, o Tcl permite estender a linguagem através da disponibilização de novos comandos, possibilitando desta forma, o suporte às mais variadas disciplinas. O desenvolvimento de novos comandos pode ser feito em Tcl ou em C.

No entanto, o Tcl tem o problema característico das linguagens interpretadas : programas feitos em Tcl executam de uma forma mais lenta que programas desenvolvidos em C ou C++. Além disso o Tcl só suporta um tipo de dados - cadeias de carácters - tornando ineficiente o tratamento de estruturas de dados complexas.

O Tcl não é uma linguagem orientada a objectos. Esta característica poderá ser considerada uma limitação quando se pretende utilizar esta linguagem no desenvolvimento de aplicações mais complexas. Esta limitação foi entretanto resolvida com a disponibilização de uma extensão designada por `[incr TCL]` [41] que proporciona o paradigma das linguagens orientadas a objectos em Tcl.

O Tcl é uma linguagem adequada para o desenvolvimento rápido de aplicações não muito complexas, o que a torna muito interessante para o desenvolvimento de protótipos. Foi esta motivação que levou à sua adopção no sistema Scotty, é essa a motivação para a sua adopção no nosso sistema.

5.1.2 O sistema Scotty/ Tkined

O Scotty é uma extensão Tcl que disponibiliza um conjunto de comandos para desenvolvimento de aplicações de controlo e gestão de uma rede e seus elementos. O Scotty disponibiliza extensões Tcl para:

- O envio e recepção de pacotes ICMP.
- A consulta de informação de DNS (Domain Name System).
- Acesso através de TCL a sockets UDP (O Tcl base só suporta TCP).
- O envio e recepção de mensagens SNMP (SNMPv1, SNMPv2USEC, SNMPv2C).
- Suporte ao desenvolvimento de agentes SNMP.
- O acesso e interpretação das definições das MIBs de SNMP.
- O suporte ao agendamento de actividades.

- Suporte à representação da rede.

O Tkined é um editor de rede interactivo que suporta o desenho de representações da rede. O Tkined pode ser utilizado como uma interface gráfica de um conjunto de aplicações que actuam sobre a rede e seus elementos. Estas aplicações executam-se em processos separados que comunicam com o Tkined através de um protocolo muito simples, encapsulado numa das extensões disponibilizadas pelo Scotty, o `ined`. Desta forma as aplicações podem actuar sobre os elementos da rede e reflectir o seu estado nas representações espelhadas pelo Tkined.

O Tkined define e representa os diferentes elementos da rede num ambiente gráfico, disponibilizando um conjunto de serviços para o tratamento dessa representação. O Scotty disponibiliza o conjunto de funções necessárias ao acesso e manipulação dos elementos de rede, podendo os programas desenvolvidos com base nesta extensão interagir com a representação da rede mantida pelo Tkined (Figura 5.2).

No Scotty, a rede e seus elementos são tipificados através dos elementos: nó (*node*), elo (*link*), rede (*network*) e grupo (*group*). Estas entidades poder ser manipuladas através de duas extensões Tcl que se encontram integradas no Scotty: `Tnm::ined` e `Tnm::map`.

Como vimos anteriormente, o `ined` está directamente relacionado com a manipulação da representação da rede mantida pelo Tkined, permitindo que os *scripts* lançados através deste editor a possam de interagir com a representação dos objectos (alteração da cor, de atributos, etc). A Figura 5.3, apresenta alguns exemplos de comandos `ined` relativos à manipulação de objectos Tkined.

A extensão `Tnm::map`, desenvolvida posteriormente, veio possibilitar a existência de aplicações que interagem sobre uma representação da rede independente do Tkined, dando ao programador a capacidade de desenvolver aplicações onde a apresentação da rede pode ser independente da sua representação.

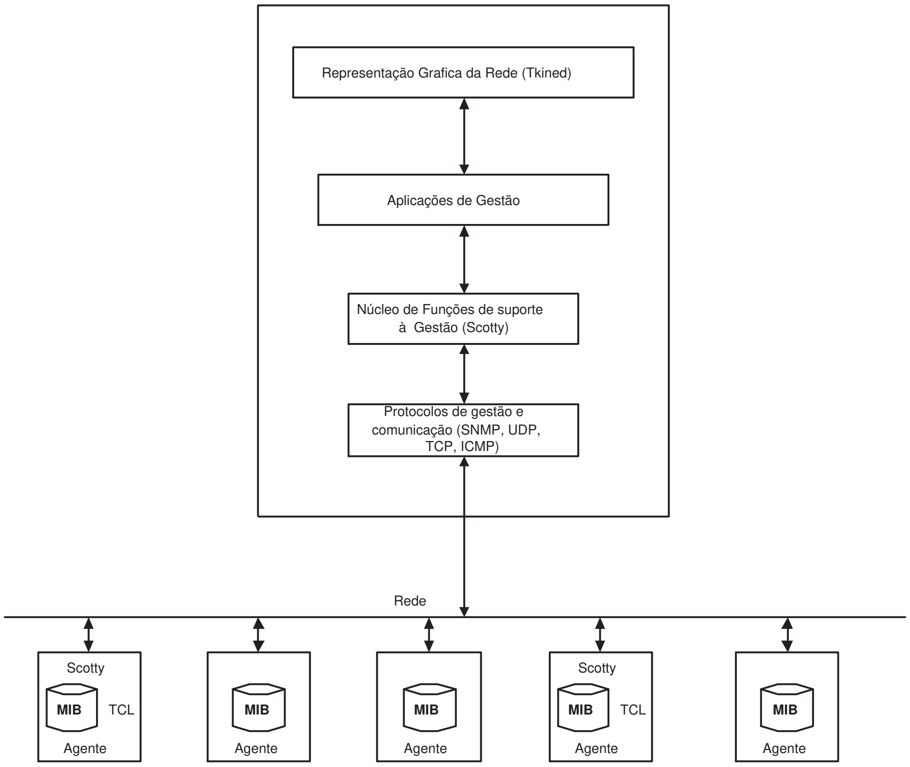

Figura 5.2: Relações entre a rede e o Tkined/Scotty

```
# Criacao de um objecto ined(item) do tipo Node
set nl [ined create NODE]

# Alteracao do nome de um item
ined name $n1 "teste1"

# Alteracao de um atributo de um item
ined attribute $n1 Localizacao "Lisboa"

# Eliminacao de um item
ined delete $n1
```

Figura 5.3: Exemplo de comando `ined`

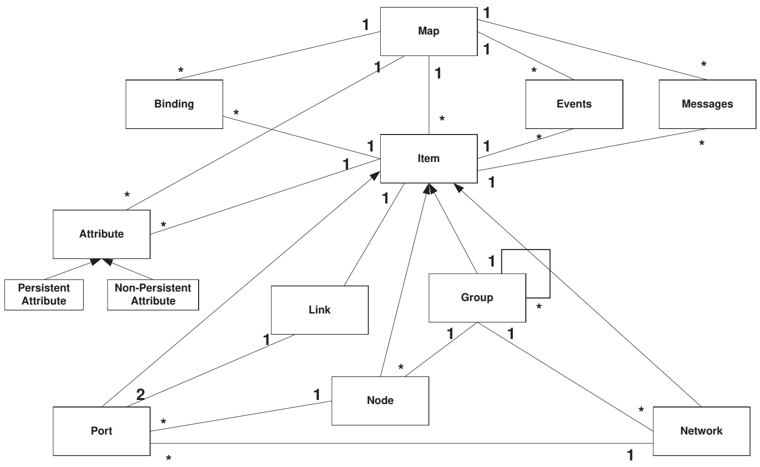

Figura 5.4: Modelo de dados do `Tnm::Map`

5.1.3 Tnm::map

O `Tnm::map` é uma extensão Tcl, integrada no sistema Scotty, que permite a manipulação de representações (mapas) da rede. Um mapa descreve a topologia de uma rede que se pretende ver gerida através de aplicações baseadas nas extensões Scotty.

Nesta extensão Tcl a representação da rede e seus elementos assenta em duas entidades: *mapa* e *item*. Um mapa descreve a topologia da rede cujos componentes são representados por items. Cada item tem associado um determinado tipo (nó, grupo, rede, porto ou elo) que define o conjunto de operações passíveis de serem realizadas sobre si.

A Figura 5.4 descreve o modelo de dados em que assenta a extensão `Tnm::Map`. Cada mapa ou item pode conter um número arbitrário de atributos definidos pelo utilizador. Existem dois tipos de atributos: atributos persistentes e atributos voláteis. Os atributos persistentes distinguem-se dos voláteis no facto de estarem incluídos no processo de salvaguarda do mapa. Os eventos voláteis perdem-se sempre que o mapa é reinicializado.

Os mapas suportam programação orientada a eventos onde é possível definir procedimentos que são executados quando ocorre um evento especifico. Cada evento é definido por um identificador (*event tag*) que é utilizado para o activar num mapa ou num item.

A associação de procedimentos ao tratamento de determinados tipos de eventos é efectuado através de um mecanismo designado por ligação (binding). A cada ligação, criada no contexto de um mapa ou de um item, é associado um padrão que está relacionado com o identificador ou identificadores dos eventos que se pretende tratar. Quando um determinado mapa ou item recebe um evento, o sistema verifica se o mesmo tem alguma ligação cujo padrão corresponda ao identificador do evento. No caso positivo o procedimento associado à ligação é executado.

Por omissão, um evento evolui de uma forma hierárquica, iniciando-se no item onde ocorreu o evento e propagando-se de forma ascendente na hierarquia de entidades, atingindo em último lugar o MAPA. O seu tratamento, pode ocorrer em cada entidade por onde um evento é propagado, caso exista alguma ligação associado ao evento em causa. Os procedimentos que tratam os eventos têm a capacidade de interromper esta propagação.

Os parâmetros do evento poderão ser passados para o procedimento, se este contiver no seu código determinadas chaves de substituição que no momento da execução do procedimento são substituídas pelas informações pretendidas.

Encontram-se definidas as seguintes substituições:

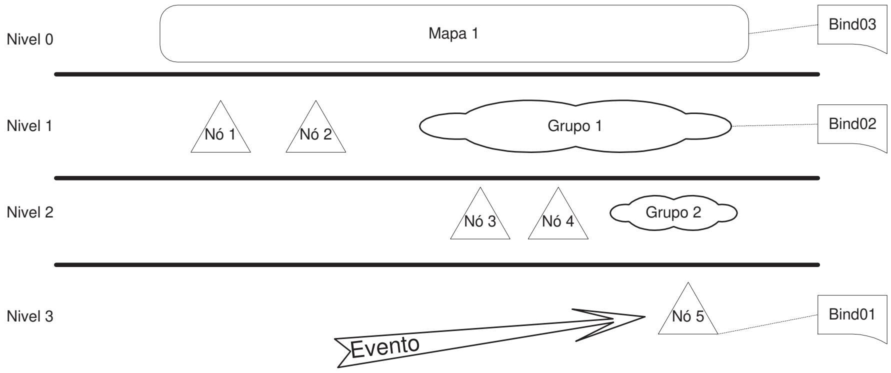

Figura 5.5: Exemplo do funcionamento do mecanismo de eventos no `Tnm::map`

%A	Parâmetros do evento.
%B	Testemunho da ligação (*binding*).
%E	Testemunho do evento
%I	Testemunho do item se o evento ocorrer num item
%M	Testemunho do mapa se o evento ocorrer num mapa
%N	Nome do evento que accionou o procedimento
%P	Padrão do Evento

Considere-se um mapa cuja hierarquia se encontra descrita na Figura 5.5, em que determinado tipo de evento é tratado ao nível do Mapa 1, do Grupo 1 e do Nó 5. Caso se verifique um evento do tipo especificado, ocorre o seu tratamento ao nível do Nó 5, do Grupo 1 e do Mapa 1. De notar que este evento é também recebido pelo Grupo 2, mas sem qualquer efeito uma vez que aquele item não tem definido qualquer tratamento do evento. A propagação do evento poderia terminar no Nó 5, caso o procedimento deste item assim o determinasse.

O `Tnm::map` dispõe ainda de um mecanismo de registo (`log`), designado por mensagens (messages), que se caracteriza pela capacidade das entidades `Tnm::map` receberem e armazenarem mensagens geradas pelas aplicações. Uma mensagem é composta por uma estampilha temporal, um intervalo de tempo, um identificador e um campo de texto. As mensagens são mantidas em memória durante um curto espaço de tempo. Podem entretanto ser armazenadas em ficheiros caso se verifiquem as seguintes condições: o mapa ou item deve ter definido um directório de armazenamento das mensagens (opção `-store`) e o identificador da mensagem deve corresponder a um dos padrões definidos para o armazenamento de mensagens. Estes padrões são definidos no mapa ou nos items usando opção -store".

A titulo de exemplo consideremos um nó cuja opção de `-path` contém "`/tmp/yourmap`" e a opção `-store` contém "`ifload`". Se for gerada uma mensagem em 2 de Abril de 2000 com o identificador "`ifload-3`" e o texto "52", esta mensagem será armazenada no ficheiro `/tmp/yourmap/2000-04-02/ifload-3`.

A extensão `Tnm::map` disponibiliza um comando com o mesmo nome que suporta a criação de mapas:

```
Tnm::map create [option value]
```

Este comando devolve um testemunho (*handler*) que suporta a manipulação do mapa. Sobre um mapa, pode-se executar um conjunto variado de operações, destacando-se as seguintes (`map#` é o testemunho do mapa):

map# attribute name value	Definição de atributos.
map# attribute name	Visualização de atributos.
map# bind pattern script	Associação de um procedimento (script) a um padrão de eventos que se pretende tratar.
map# cget option	Visualização das opções do mapa. Por exemplo:
-name name	Nome do mapa
-expire secs	Tempo de vida das mensagens em memória.
-path path	Directoria de armazenamento das mensagens.
-store pattern	Padrões das mensagens que devem ser armazenadas.

map# clear	Inicialização do mapa.
map# configure [option value]	Configuração das opções do mapa.
map# copy items	Cópia de items de um mapa.
map# create type [option value]	Criação de items. O parâmetro type pode corresponder a `node`, `group`, `port`, `network`, `link`.
map# destroy	Eliminação do mapa.
map# find [options value]	Função de consulta do mapa.
map# load channel	Carregamento do conteúdo de um mapa.
map# message tag text	Envio de uma mensagem para o mapa.
map# paste	Inserir no mapa items previamente copiados.
map# raise tag [args]	Activação de um evento.
map# save channel	Salvaguarda do mapa através de um canal.

Os items são acedidos através do testemunho que é devolvido pelo sistema no acto de criação. As operações passiveis de serem executadas sobre os items têm muitas analogias com os comandos que podem ser executados sobre os mapas:

item# attribute name value	Atribuição de atributos.
item# attribute name	Visualização de atributos.
item# bind pattern script	Associação de um procedimento (script) a um determinado padrão de eventos que se pretende tratar.

item# cget option	Visualização das opções do item. Destacam-se as seguintes opções:
-name name	Nome do item.
-color color	Cor do item.
-ctime	Data e hora de criação do item.
-expire secs	Tempo de vida das mensagens em memória.
-group group	Associação do item a um grupo.
-path path	Directoria de armazenamento das mensagens.
-store pattern	Padrões das mensagens a armazenar.
-tags	Identificadores que associam o item a determinados grupos de elementos da rede (Exemplo: encaminhadores, comutadores, impressoras, etc).
item# configure [option value]	Configuração das opções do item.
item# destroy	Eliminação do item.
item# map	Devolve o mapa que se encontra associado ao item.
item# message tag text	Envio de uma mensagem para o item.
item# raise tag [args]	Activação de evento no item.
item# type	Devolve o tipo de item (*node*, *group*, *link*, *network*, *port*).

Vimos anteriormente que as diferentes entidades do Tnm::map poderiam se caracterizadas por atributos persistentes e não persistentes. Estes dois tipos de atributos distinguem-se pelo nome. Os nomes dos atributos não persistentes podem ser constituídos por quaisquer letras ou números desde que o primeiro caracter não corresponda a uma letra maiúscula nem se iniciem com o caracter ":".

O nome dos atributos persistentes é normalmente iniciado pelo nome do âmbito ou contexto do atributo seguido do seu nome e terminado com vários qualificadores. O caracter “:“ é utilizado na separação entre as diferentes componentes do nome do atributo. Cada uma das componentes do nome deverá ser iniciada por uma letra maiúscula. Se seu nome é iniciado com o caracter “:“ o atributo não está associado a um âmbito, dizendo-se que é global. Vejamos alguns exemplos:

Contacto:Nome:Apelido	Nome de um atributo persistente com um ambito associado
:Localizacao:Rua	Nome de um atributo persistente global
teste:testea	Nome de um atributo não persistente

Sempre que é activado um evento num item ou num mapa, o sistema cria um objecto devolvendo o seu testemunho por forma a que seja possível a realização de posteriores tratamentos do evento. Sobre um evento podem-se efectuar as seguintes operações:

event# args	Visualização dos parâmetros que foram passados na activação do evento.
event# destroy	Destruição do objecto evento.
event# item	Se o alvo do evento for um item então devolve o seu testemunho respectivo.
event# map	Se o alvo do evento for um mapa então devolve o seu testemunho respectivo.
event# tag	Devolve o identificador associado ao evento.
event# time	Data e hora da geração do evento.

As mensagens sofrem de tratamento idêntico aos eventos:

message# destroy	Destruição do objecto mensagem.
message# item	Se o alvo da mensagem for um item então devolve o seu testemunho respectivo.
message# map	Se o alvo da mensagem for um mapa então devolve o seu testemunho respectivo .
message# tag	Devolve o identificador associado à mensagem.
message# text	Devolve o texto associado à mensagem.
message# time	Data e hora da geração da mensagem.

Por último temos o tratamento das ligacões (Bindings). Analogamente aos eventos e as mensagens, sempre que é efectuado uma ligação, o sistema cria um objecto devolvendo um testemunho o que possibilita desta forma o seu tratamento:

bind# destroy	Destruição do objecto.
bind# item	Se a ligação estiver associada a um item então devolve o seu respectivo testemunho.
bind# map	Se a ligação estiver associada a um mapa então devolve o seu respectivo testemunho.
bind# pattern	Devolve o padrão associado à ligação.
bind# script	Devolve o procedimento associado à ligação.

A Figura 5.6 fornece alguns exemplos da utilização da extensão `Tnm::map`.

5.2 Concretização no modelo de vistas proposto

Como vimos, o sistema Scotty/Tkined já dispõe de duas extensões Tcl destinadas à representação da rede e dos seus elementos (`ined` e `map`). O desenvolvimento de uma nova extensão obrigaria os seus utilizadores por um lado a aprenderem novos coman-

```
# Criacao de um mapa
set m1 [Tnm::map create]

# Criacao de um no de um mapa e
# colocacao do seu testemhunho em n1
set n1 [$m1 create node]

#Alteracao do nome do no
$n1 configure -name "teste1"

# Eliminacao de um no
$n1 destroy
```

Figura 5.6: Exemplos de comando `map`

dos e por outro a rescreverem os programas que utilizam as actuais extensões. Para evitar esta desvantagem, optou-se por acrescentar o nosso modelo à extensão `Tnm::map`, possibilitando desta forma a convivência de programas que utilizam a extensão original ou o modelo por nós estendido.

5.2.1 Extensão `Tnm::repo`

A extensão `Tnm::repo` ("repo" de repositório) é uma expansão à extensão `map` com o objectivo de acrescentar a esta algumas funcionalidades, donde se destaca o mecanismo de vistas anteriormente apresentado. A Figura 5.7 descreve o modelo de dados desta extensão.

Verificam-se consideráveis diferenças entre o `Tnm::repo` e o `Tnm::map`. No `Tnm::repo`, a entidade responsável pela agregação e armazenamento de todas as entidades envolvidas na representação da rede é o repositório (REPO). Num repositório são mantidos os dispositivos, métodos e vistas. No `Tnm::map` a função de armazenamento das entidades envolvidas na representação da rede é suportada pela entidade MAP. Existem claras diferenças entre o papel do MAP e o papel do REPO: No `Tnm::repo` as responsabilidades da entidade MAP foram repartidas com a entidade REPO, distinguindo desta forma a representação dos elementos da rede (cuja responsabilidade foi atribuída à enti-

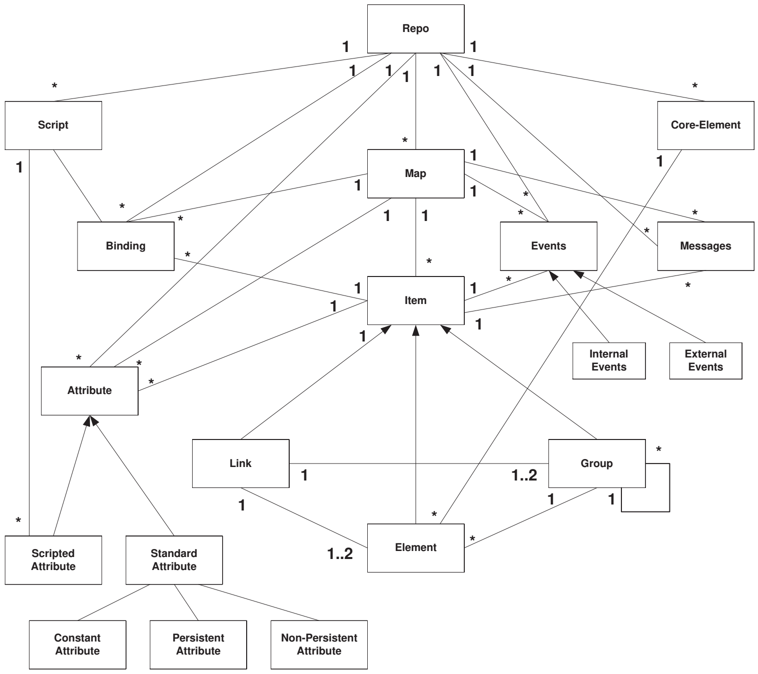

Figura 5.7: Modelo de dados do `Tnm::repo`

dade REPO) da representação das relações entre os elementos da rede, (responsabilidade assumida pela entidade MAP que representa uma VISTA). De igual modo a representação dos elementos da rede sofre algumas alterações no `Tnm::repo` relativamente ao `Tnm::map`.

No `Tnm::map` um elemento da rede é representado por um item que se encontra referenciado num MAP. Um elemento da rede pode ser tipificado como nó (*node*), elo (*link*), rede (*network*) ou porta (*port*).

No `Tnm::repo`, cada elemento da rede tem uma representação unívoca no repositório designada por dispositivo que reflecte o estado do elemento da rede. Um dispositivo, ou elemento-nuclear, designado em inglês por “Core-Element” (CELEM), não mantém qualquer informação sobre as suas relações com outros dispositivos. Essa informação é mantida ao nível das vistas (MAPS) pelos seus embaixadores.

No `Tnm::repo` deixa de existir a representação rígida dos elementos da rede que se verifica no `Tnm::map`. No nosso sistema um item pode ser um embaixador (designado em inglês por "element" - ELEM), um elo (LINK) ou uma pasta (GROUP). Uma pasta pode ser vista como um elemento agregador de outras entidades ou uma entidade mais complexa como uma rede ou um elemento da rede que agrega vários componentes. Por exemplo, um computador pessoal pode ter diferentes representações conforme as necessidades de refinamento da representação (Figura 5.8):

Alternativa A: Um DISPOSITIVO referenciado num repositório e um ou mais EMBAIXADORES referenciados em uma ou mais VISTAS.

Alternativa B: Vários DISPOSITIVOS que representam os diferentes componentes do computador pessoal (disco, processador, memória, teclado, porta série, placa ethernet, etc.) e vários EMBAIXADORES agregados numa VISTA, representando a PASTA o computador pessoal.

No `Tnm::repo`, não se condiciona a representação dos elementos da rede a um conjunto tipificado de entidades, disponibilizando-se apenas os elementos necessários ao estabelecimento das relações entre eles:

Embaixadores (Elements) : Representantes de elementos concretos da rede (físicos ou lógicos).

Elos (links) : Representantes das relações de ligação entre elementos da rede.

Pastas (groups) : Representantes das relações de inclusão entre elementos da rede.

Verificam-se também algumas diferenças entre o `Tnm::repo` e o `Tnm::map` na caracterização dos atributos. No `Tnm::map`, os atributos só poder ser tipificados pela sua persistência (Persistentes ou não persistentes). No `Tnm::repo`, os atributos têm mais

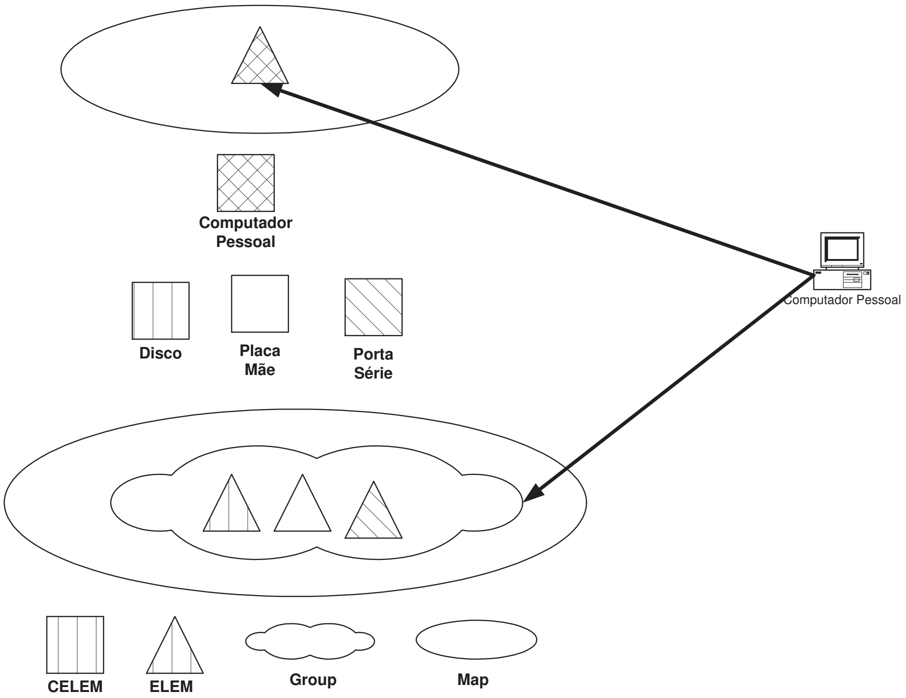

Figura 5.8: Exemplo da capacidade de representação multipla no `Tnm::repo`

um nível de tipificação: a sua complexidade. Assim temos atributos simples e métodos. Os atributos simples podem ainda ser tipificados com atributos persistentes, não persistentes ou constantes.

Os métodos (*Scripted Attributes*) são atributos associados a procedimentos que são mantidos ao nível do repositório. Foram seguidos os princípios que se encontram presentes no tratamento dos eventos do `Tnm::map`. A evocação de um atributo que esteja associado a um procedimento atribuirá ao valor deste o resultado da execução do procedimento. A Entidade REPO é responsável por manter o código dos procedimentos o qual poderá ser utilizado e referenciado em um ou mais atributos ou em uma ou mais ligações(Bindings), permitindo desta forma a reutilização de código.

Às variantes persistentes e não persistentes que se encontravam disponiveis nos atributos simples, foram adicionados os atributos constantes. Uma funcionalidade incluída no `Tnm::repo` corresponde à capacidade de clonagem de dispositivos ou items, como forma de reduzir os problemas que advêm do facto deste sistema não ser orientado a objectos. Da clonagem de um dispositivo ou item, resulta um novo elemento que mantem os mesmos atributos e tratamento de eventos do elemento origem. No entanto, o conteúdo dos atributos não é copiado excepto nos atributos constantes. Os atributos constantes permitem definir informações que se devem manter inalteradas nos diferentes dispositivos ou items que foram criados através do processo de clonagem.

Por fim verificaram-se alterações ao nível do tratamento de eventos. O `Tnm::map` só permite o tratamento de eventos externos. No `Tnm::repo` adicionalmente aos eventos externos é possível tratar os eventos internos, ou seja, eventos que reflectem acontecimentos associados à manipulação das diferentes entidades do repositório. Esta capacidade é fundamental para a caracterização dos dispositivos e items. Consideremos o exemplo anteriormente referido, da representação de um computador pessoal através de um grupo. Sem o tratamento de eventos internos, nada impede que se inclua no grupo que representa um computador pessoal, um DISPOSITIVO que represente, por exemplo, uma impressora. O tratamento dos eventos associados à inclusão de items num grupo torna a representação mais coerente, uma vez que o utilizador dispõe de um mecanismo que lhe permite validar as operações efectuadas.

5.2.2 Interface da Extensão REPO

São descritos de seguida, os principais aspectos da interface que concretiza o modelo de vistas no Scotty. De notar que, tendo em conta o âmbito deste documento, terá pouco interesse efectuar uma descrição exaustiva da interface, pelo que será apresentado com o grau de pormenor suficiente para a sua compreensão. O manual do Tnm::repo [53] descreve mais em promenor a interface e funcionalidades da extensão. Nesta descrição será dada especial ênfase às particularidades da interface que o distinguem do `Tnm::map`.

Actividades de criação

A concretização do modelo anteriormente apresentado assenta na interface `Tnm::repo`. Este comando Tcl permite a criação de repositórios:

```
Tnm::repo create [option value]
```

Este comando devolve um testemunho (*handler*) que suporta o conjunto de operações passíveis de serem efectuadas sobre o repositório. As opções possíveis caracterizam atributos que encontramos no `Tnm::map`:

-name	nome do repositório
-path	Directoria de armazenamento das mensagens
-store	Padrões das mensagens que devem ser armazenadas
-expire	Tempo de Vida das mensagens em memória

O repositório é responsável por manter toda a informação relativa ao representantes dos elementos da rede (DISPOSITIVOS - CELEM) e das múltiplas relações em que estes podem estar envolvidos (VISTAS - MAP).

O repositório está directamente relacionado com três componentes da representação: dispositivos, vistas e métodos. A inserção dos métodos ao nível do repositório tem a vantagem destes poderem ser referenciados por todas as restantes entidades de que se compõe o repositório.

Directamente sobre um repositório podem-se criar dispositivos (CELEMS), vistas (MAPS) e métodos (SCRIPTS), através dos seguintes comandos:

```
repo# create celem [option value]
repo# create map [option value]
repo# create script [option value]
```

Da execução de qualquer destes comandos, resulta a devolução de um testemunho que permite o acesso posterior à entidade criada.

Em cada vista é mantida a informação relativa à representação de uma realidade específica do universo representado pelo repositório. A vista é directamente responsável por manter informação sobre aos seus embaixadores (ELEM), pastas (GROUP) e elos (LINK). É através do testemunho de uma vista que são criadas estas entidades.

```
map# create elem celem-handler [option value]
map# create group [option value]
map# create link [option value]
```

Estes comandos representam a forma explícita de criação de entidades associadas à vista. Os embaixadores podem ainda ser criados de uma forma implícita, que consiste na associação à vista de um filtro que define quais as propriedades que cada DSPOSITIVO deverá ter para que os seus embaixadores possam ser incluidos na vista. Na presença de um filtro, o `Tnm::map` efectua automaticamente a criação/manutenção dos embaixadores da vista. O filtro corresponde a uma lista de atributos e respectivos valores que é associada à opção `-filter`. A caracterização da vista de uma forma implícita só é possivel de ser efectuada se esta ainda não tiver associado nenhum embaixador.

```
repo#create map -filter [list attribute==value]*
```

Vejamos um exemplo de criação de repositório contendo a representação de uma rede composta por dois computadores pessoais, duas impressoras e três vistas: Um vista contendo todos os elementos da rede e mais duas vistas de agregação dos computadores pessoais e impressoras.

```
# Criação do repositório
set repo1 [Tnm::repo create -name repo1]

#Criação dos dispositivos
set pc001 [$repo1 create celem -subtype pc -name pc001]
set pc002 [$repo1 create celem -subtype pc -name pc002]
set pr001 [$repo1 create celem -subtype printer -name pr001]
set pr002 [$repo1 create celem -subtype printer -name pr002]

#Definicao de atributos
$pc001 attribute set :MachineType "pc"
$pc002 attribute set :MachineType "pc"
$pr001 attribute set :MachineType "printer"
$pr002 attribute set :MachineType "printer"

# Criação da vistas explicitas
set view1 [$repo1 create map -name allelements]
set view2 [$repo1 create map -name allpc]
set view3 [$repo1 create map -name allprinters]

# Criação da vistas implicitas identicas às vistas explicitas
set view4 [$repo1 create map -name Iallelements -filter [list :MachineType != ""]]
set view5 [$repo1 create map -name Iallpc -filter [list :MachineType == pc]]
set view6 [$repo1 create map -name Iallprinters -filter [list :MachineType == printer]]

# Criação dos embaixadores
set elem001 [$view1 create elem  $pc001 -name pc001]
set elem002 [$view1 create elem  $pc002 -name pc002]
set elem003 [$view1 create elem  $pr001 -name pr001]
set elem004 [$view1 create elem  $pr002 -name pr002]
set elem005 [$view2 create elem  $pc001 -name pc001]
set elem006 [$view2 create elem  $pc002 -name pc002]
set elem007 [$view3 create elem  $pr001 -name pr001]
set elem008 [$view3 create elem  $pr002 -name pr002]
```

De notar que não é permitida a criação de embaixadores em vistas implicitas uma vez que esta tarefa é executada pelo `Tnm::map` com base no filtro definido. Por questões de desempenho da extensão, a actualização das vistas implicitas não é efectuada automaticamente, sendo necessário executar o seguinte comando:

```
map# update
```

Tratamento de atributos

A manipulação dos atributos definidos pelo utilizador assenta nas três operações seguintes:

```
handler# attribute set attribute-name [value | options value]
handler# attribute get attribute-name [options]
handler# attribute unset attribute-name
```

No `Tnm::map`, a criação de um atributo resume-se à associação do nome do atributo ao seu respectivo valor. A eliminação de um atributo corresponde à atribuição de uma cadeia vazia ao atributo:

```
# Criação de um atributo
$s attribute teste1 "teste"
$ Remoção de um atributo
$s attribute teste1 ""
```

Este procedimento embora de simples utilização não nos parece adequado a algumas necessidades da nossa interface, nomeadamente em operações de clonagem de entidades onde a estrutura de atributos é copiada para a nova entidade mas sem qualquer conteúdo. Além disso, o processo de eliminação do `Tnm::map` torna-se pouco intuitivo para a eliminação de métodos. Daí que a nossa interface tenha primitivas explicitas para a manipulação de atributos.

O `Tnm::repo` segue a convenção de nomes do `Tnm::map` à qual foi acrescentada a capacidade de identificar atributos constantes: atributos cujo nome é totalmente composto por letras maiusculas. A distinção entre atributos simples e métodos é materializada nas opções de criação do atributo:

```
$o atribute set name -script script-handler -expire seconds -poll yes
```

A definição do método é efectuada a nível do repositório, permitindo a sua reutilização. A associação do atributo ao método é efectuada através da opção `-script` pela inclusão do respectivo testemunho. A opção `-expire` permite o controlo de validade da informação processada pelo procedimento. Caso o atributo seja consultado com o seu valor expirado, o sistema força a execução do procedimento para a disponibilizar um valor do atributo actualizado. A opção `-poll` distingue a estratégia de actualização da informação: quando expira o tempo pré-definido entre execuções do procedimento ou no momento da consulta.

Os pormenores do contexto que envolve a execução do procedimento podem ser conhecidos, se este conviter no seu código determinadas chaves de substituição que, em tempo de execução, são substituídas pelas informações pretendidas. Encontram-se definidas as seguintes substituições:

%V	Nome do atributo.
%L	Valor actual do atributo.
%T	Tipo de entidade que se encontra associada ao procedimento. Valores possíveis:
- repo	Repositório
- celem	Dispositivo
- map	Vista
- elem	Embaixador
- link	Elo
- group	Pasta

%H	Testemunho da Entidade associada ao procedimento.
%O	Operação efectuada sobre o atributo. Valores possíveis:
- get	Operação de leitura.
- set	Operação de escrita.
- unset	Operação de eliminação.
- raiseup	Evento Ascendente.
- raisedown	Evento descendente.
- raiseside	Evento lateral.
%S	Testemunho do procedimento.
%R	Testemunho do Repositório.
%A	Parâmetros do evento.
%B	Testemunho da ligação.
%E	Testemunho do evento se for evento
%N	Nome do evento que accionou o procedimento
%P	Padrão do Evento

O valor que é devolvido quando se acede a um método é definido no procedimento através da inserção de uma instrução de atribuição a uma variável cujo nome deverá ser idêntico ao nome do procedimento. Vejamos o seguinte exemplo:

```
repo0 create script -name val2 -script {
        set val2 2
}
```

A especificação de uma opção numa evocação da operação `get` sobre um atributo, tem como resultado a obtenção do valor da opção para o atributo em causa. Por exemplo o testemunho do procedimento associado a um método pode ser obtido através do seguinte comando:

```
handler# attribute get attribute-name -script
```

Tratamento de Eventos

O tratamento de eventos segue também, no essencial, a estratégia adoptada no `Tnm::map`:

```
handler# bind tag-pattern -script script-handler [-down | -up | -side | -all]
```

Caso ocorra um evento cujo nome esteja associado num determinado objecto a um método, verifica-se a sua execução.

As opções `-up`, `-down` e `-side` permitem distinguir diferentes processamentos associados aos diferentes sentidos de propagação dos eventos.

A ocorrência de um evento materializa-se na primitiva:

```
handler# raise [-up | -down | -side[hops]] tag [args]
```

Os eventos internos, reconhecidos através de um conjunto de identificadores pré-definidos, não podem ser accionados externamente à entidade, o que torna inválidos os seguintes comandos:

```
$r raise -up :Repo:InternalEvents:CreateCelem
$r raise -down :Item:InternalEvents:SetAttribute
$r raise -up :View:InternalEvents:CreateElem
```

Os pormenores do contexto associado ao evento são transferidos para o procedimento de forma análoga à à utilizada no tratamento de eventos do `Tnm::map` e na evocação de métodos. Assim, mantem-se o mecanismo de chaves de substituição vigente para os métodos.

Obtenção de informações

O comando `info` aplicado a qualquer dos elementos desta extensão, permite obter um conjunto variado de informações. Vejamos alguns exemplos:

Sobre um repositório:

repo# info subject ?pattern?	Disponibiliza a lista de testemunhos relacionados com a entidade que cumpram o sujeito e o padrão definido pelo utilizador. Os sujeitos possíveis são:
- celems:	Lista de dispositivos associados ao repositório.
- maps:	Lista de mapas associados ao repositório.
- scripts:	Lista de métodos mantidos pelo repositório.
- attributes:	Lista de nomes dos attributos associados ao repositório.
- events:	Lista dos eventos que ocorreram no repositório.
- bindings:	Lista das associações de tratamento de eventos.
- messages:	Lista das mensagem enviadas ao repositório.

Sobre o dispositivo:

celem# info attributes ?pattern?	Devolve a lista de nomes dos atributos.
celem# info elems ?pattern?	Devolve a lista de embaixadores associados ao dispositivo.
celem# info bindings ?pattern?	Devolve a lista de ligações de tratamento de eventos.
celem# info events ?pattern?	Devolve a lista de eventos que ocorreram no dispositivo.
celem# info messages ?pattern?	Devolve a lista de mensagens enviadas ao dispositivo.

Sobre a vista:

map# info attributes ?pattern?	Devolve a lista de nomes dos atributos
map# info elems ?pattern?	Devolve a lista de Embaixadores associados à vista.
map# info groups ?pattern?	Devolve a lista de grupos associados à vista.
map# info links ?pattern?	Devolve a lista de elos associados à vista.
map# info toplevel	Devolve os elementos pertencentes ao primeiro nível hierárquico de uma vista (na prática os elementos sem pai de uma vista).
map# info bindings ?pattern?	Devolve a lista de ligações de tratamento de eventos.
map# info events ?pattern?	Devolve a lista de eventos que ocorreram sobre a vista.
map# info messages ?pattern?	Devolve a lista de mensagens enviadas à vista.

Sobre um item:

item# info attributes ?pattern?	Devolve a lista de nomes dos atributos.
item# info bindings ?pattern?	Devolve a lista de ligações de tratamento de eventos.
item# info events ?pattern?	Devolve a lista de eventos que ocorreram sobre o item.
item# info messages ?pattern?	Devolve a lista de mensagens enviadas ao item.
item# info neighbors	Devolve a lista de items vizinhos.

Operações de manutenção

Nas operações de manutenção incluem-se os comandos de carregamento e salvaguarda, de destruição e comandos de cópia e clonagem dos elementos constantes do repositório:

repo# save ?channel?	Efectua uma salvaguarda do repositório para o canal (testemunho de um ficheiro, um socket,etc) especificado.
repo# load ?channel?	Carrega para o repositório a salvaguarda que se encontra disponivel no canal especificado.
repo# copy celem	Coloca no "clipboard" uma cópia do elemento-nuclear.
repo# paste	Inclui no repositório as entidades constantes do "clipboard".
repo# dump	Devolve uma cópia do repositório.
repo# destroy	Elimina o repositório.
celem# clone	Efectua uma operação de clonagem de um dispositivo.

Operações especializadas

Existem algumas operações especificas a determinadas entidades. Distinguem-se em opções e comandos. Analogamente ao `Tnm::map`, a manipulação das opções pode ser efectuada através dos comandos `configure` e `cget`. Vejamos alguns exemplos aplicados a items:

item# configure -group $f	Especifica o item pai da entidade em causa.
item# cget -group	Devolve o item pai da entidade especificada.
item# configure -subtype router	Especifica o subtipo do item.
item# cget -subtype	Devolve o subtipo do item.

Estes comandos estão normalmente associados a operações de leitura de atributos do elemento.

item# map	Devolve o testemunho da vista associada a uma entidade (válido para embaixadores, pastas e elos).
item# repo	Devolve o testemunho do repositório associado à entidade.
map# repo	Devolve o testemunho do repositório associado à vista.

Restrições

As restrições materializam-se no estabelecimento de listas de acesso ao nível das vistas:

```
map# configure -constraints ?list?
```

A lista deverá ser constituída por um conjunto de regras que estabelecem qual a informação das entidades que se encontra visível através da vista:

```
Permit :Counters:X25
Deny :Generalinf
```

As regras são verificadas sequencialmente. O processo de validação de um atributo termina na primeira regra cujo critério corresponda ao nome do atributo.

Sumário

Este Capítulo descreve uma concretização particular do modelo proposto no capítulo anterior que se materializou numa extensão Tcl designada por Tnm::repo. O Tnm::repo baseou a sua interface na extensão Tnm::map, outra extensão Tcl que integra a plataforma Tnm/Scotty/Tkined com um objectivo identico: representar uma rede e os seus elementos.

A utilização da extensão Tnm::map como base de trabalho ao desenvolvimento do Tnm::repo teve como motivações reduzir o tempo de aprendizagem para a utilização da extensão, permitir a co-habitação de procedimentos desenvolvidos em Tnm::map, possibilitar a integração simples e intuitiva da extensão na plataforma Tnm/Scotty e obviamente, simplificar o desenvolvimento do nosso protótipo. Além dos conceitos descritos no modelo, foram incorporados no Tnm::repo um conjunto de facilidades adicionais, destacando-se a capacidade de clonagem de elementos e integração no repositório dos "scripts"passiveis de serem utilizados no tratamento de eventos e na definição de métodos.

Capítulo 6

Avaliação

A principal motivação que levou ao desenvolvimento do `Tnm::repo` foi a necessidade de criar um sistema de gestão que permitisse o tratamento coerente de múltiplas representações de uma rede heterogénea. A avaliação do sistema proposto nesta dissertação pode ser feita sobre duas vertentes distintas: numa perspectiva de funcionalidade do sistema e em termos do seu desempenho.

O método de avaliação adoptado assentou na comparação do nosso sistema com alguns dos sistemas de gestão de rede anteriormente descritos. Optou-se por comparar o `Tnm::repo` com o `Tnm::map` e com o *HP OpenView* (HPOV). Deste modo, ilustram-se as vantagens comparativas que o nosso sistema apresenta em relação ao sistema que lhe serviu de base e em relação a um dos sistemas comerciais de gestão de rede mais utilizados. Para a realização dos testes de avaliação foi utilizado o seguinte conjunto de recursos:

- Uma infra-estrutura de rede heterogénea como alvo de referência para aplicação dos sistemas de gestão (ver a secção seguinte).

- Recursos de suporte aos sistemas `Tnm::Map` e `Tnm::repo`: Um Computador Pessoal com um processador Pentium III a 450 Mhz e 128 Mb de memória; Sistema

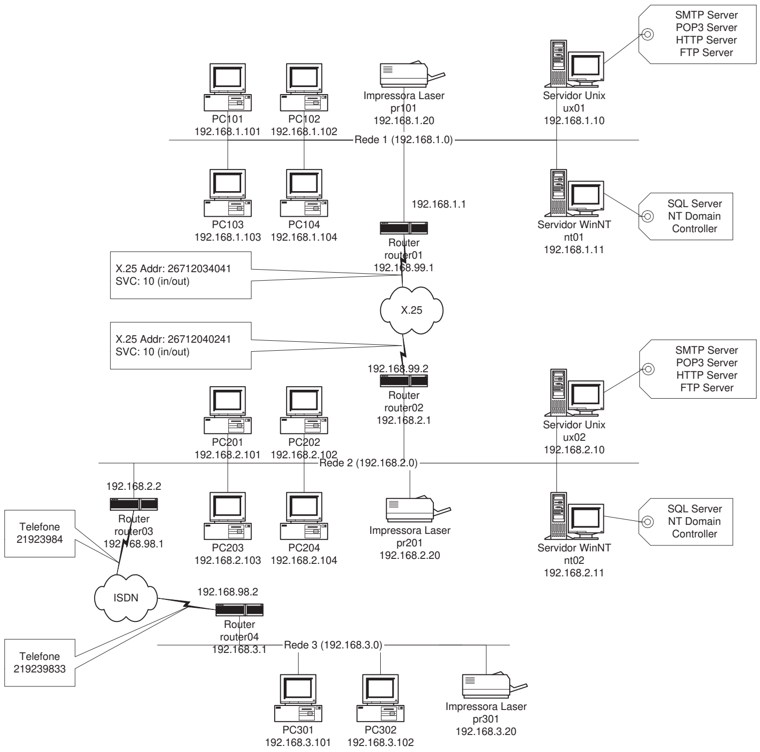

Figura 6.1: Infra-Estrutura de rede utilizada nos testes de avaliação

Operativo Linux Red Hat 5.2; Scotty versão 3.0.0; `Tnm::repo` versão 0.7.0.

- Recursos de suporte ao sistema HP OpenView: Um Computador Pessoal com um processador Pentium III a 450 Mhz e 128 Mb de memória; Sistema Operativo Windows NT 4.0 (SP 5); HP Network Node Manager Versão 5.1

6.1 Rede a gerir

A base do trabalho de avaliação assentou na gestão da rede que se encontra descrita na Figura 6.1. Esta rede é um subconjunto de uma infra-estrutura de rede vigente numa organização empresarial. A sua gestão deverá ter em conta o seguinte conjunto de requisitos:

- Existem três áreas de gestão distintas: Gestão de Rede, Gestão do ambiente Microsoft Windows e gestão dos sistemas Unix e suas respectivas aplicações.

- Os gestores da rede devem garantir a conectividade entre os diferentes sistemas e garantir a disponibilidade de toda a infra-estrutura de comunicações.

- Os gestores dos ambientes Windows devem garantir o bom funcionamento dos computadores pessoais e servidores.

- Os gestores dos sistemas Unix assegurar o funcionamento dos sistemas e aplicações que se encontram a seu cargo.

- Estas três equipas de gestão têm necessidades distintas para a gestão dos seus respectivos elementos da rede.

- A representação da rede necessária aos gestores da rede deverá responder às seguintes necessidades: $i)$ Conhecimento de todos os elementos da rede e suas respectivas interligações; $ii)$ Cada elemento da rede deverá conter os seguintes atributos genéricos: Nome do equipamento, endereço IP do equipamento, comunidade SNMP (Por questões de simplificação considera-se somente SNMPv1), localização, departamento, telefone mais próximo, utilizador responsável, tomada no sub-sistema horizontal, nome do HUB/Switch e porta do HUB/Switch, $iii)$ Os equipamentos de encaminhamento que suportem interligações em ISDN deverá conter ainda o número de telefone do respectivo acesso ISDN; $iv)$ Os equipamentos de encaminhamento deverão conter a representação das diferentes portas que dispõem, devendo cada porta conter os seguintes atributos: Tipo de porta (serie, ethernet, ISDN), protocolo de Ligação/Rede (X.25, ISDN, Ethernet), velocidade do acesso, endereço IP, endereço WAN (X.25 ou nº de telefone), número de canais adicionais, percentagem média de utilização da porta no ultimo minuto, percentagem média de utilização da porta nos ultimos 5 minutos, número de erros nos últimos 30 minutos e o número de reinicializações da ligação nas últimas 24 horas.

- Os gestores dos sistemas Windows têm as seguintes necessidades: i) Conhecimento da topologia da rede, da distribuição dos computadores pessoais e servidores pelos diferentes segmentos de rede, ii) Conhecimento do estado dos computadores pessoais e servidores. iii) Conhecimento do estado dos encaminhadores por omissão[1] de cada segmento. iv) Conhecimento nos computadores pessoais (clientes e servidores) dos seguintes atributos: Nome, endereço IP, localização, departamento, telefone mais próximo, nome do Utilizador responsável, domínio NT associado ao sistema, características do computador pessoal (processador, memória, espaço em disco, placa gráfica, interfaces de rede, etc), características do sistema operativo (sistema operativo, versão, actualizações aplicadas) e aplicações instaladas recursos que exporta (áreas de disco, impressoras, discos, etc); v) Conhecimento nos servidores dos seguintes atributos adicionais: Tipo de servidor ("stand-alone", "primary domain controller", "backup domain controler"e utilizadores locais.

- Os gestores dos sistemas Unix têm as seguintes necessidades: i) Conhecimento dos servidores Unix disponíveis e dos respectivos atributos: Nome, endereço IP, localização, características do sistema (processador, memória, espaço em disco, etc), características do sistema operativo (sistema operativo, versão, actualizações aplicadas), aplicações instaladas, serviços activos e seu respectivo estado.

De notar, que a informação que é considerada como necessária à gestão dos diferentes ambientes em teste é, obviamente, insuficiente num ambiente de gestão real. Pretendeu-se nestes requisitos dar apenas a abrangência necessária à realização da avaliação. Com os requisitos apresentados pretende-se verificar o comportamento de cada

[1]Do Inglês, "default-gateways".

um dos sistemas em estudo. Não se pretende caracterizar exaustivamente a gestão de um sistema informação.

6.2 Avaliação de funcionalidades

A avaliação das funcionalidades disponibilizadas pelos três sistemas em estudo, teve em consideração a forma como cada uma representa a rede, os seus elementos e o comportamento destes. Para se obter um termo de comparação entre sistemas de gestão, foram definidos os seguintes critérios de análise:

- *Representação da rede*: Qual a vista que é apresentada a cada administrador relativamente aos recursos que tem de gerir.
- *Tratamento da informação comum*: Qual o comportamento do sistema perante a necessidade de existirem elementos comuns a diferentes vistas.
- *Grau de abstracção com os protocolos de gestão de rede*: Que noção deverá o utilizador ter dos protocolos de gestão quando actua sobre os elementos da rede.
- *Tratamento de eventos*: Qual o modelo de tratamento de eventos disponibilizado pelos sistemas de gestão em estudo.

Numa análise superficial, poderá concluir-se que não é possível comparar o sistema HP OpenView com o `Tnm::map` e `Tnm::repo`. O OpenView é uma bancada dedicada à gestão de sistemas de informação com um especial destaque para a gestão de rede e seus dispositivos. O `Tnm::map` e o `Tnm::repo` são extensões Tcl dedicadas unicamente à representação da rede e seus elementos. No entanto, a comparação é possível quando centrada na forma como cada um dos sistemas em estudo representa as redes.

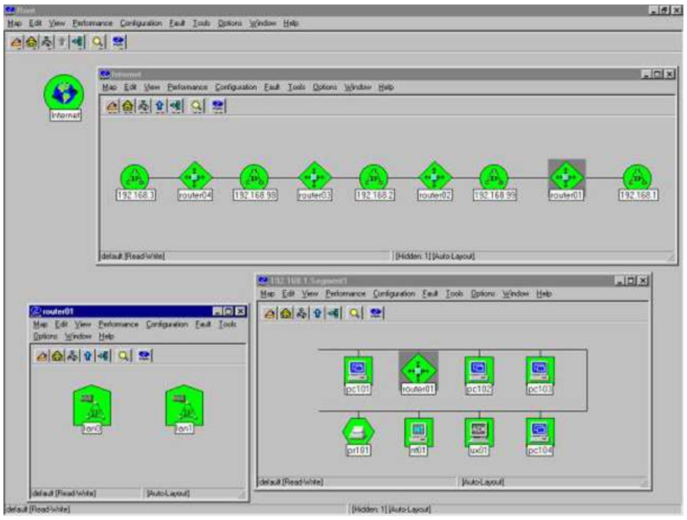

Figura 6.2: Resultado do processo de auto-descoberta do HP OpenView

6.2.1 Concretização em HP OpenView

A representação inicial da rede em estudo foi muito simples de concretizar uma vez que o HPOV dispõe de um mecanismo de auto descoberta da rede. Esta função percorre o espaço de endereçamento IP de cada um dos segmentos de rede em busca de novos dispositivos. A informação recolhida pelo serviço `netmon` é enviada para outro serviço, o `ovtopmd` (OpenView Topology Manager Service for IP discovery) que é responsável por manter a topologia da rede. A informação da rede é mantida numa base de dados, gerida por um serviço designado por `ovwdb`. A apresentação gráfica da rede ao utilizador é uma função do programa `ovw` (OpenView Windows) que interage com o serviço `ovtopmd`.

A representação resultante do processo de auto descoberta agrupa os elementos da rede pelos seus segmentos de rede IP, como se poderá verificar na Figura 6.2. Esta representação encontra-se adaptada às necessidades que foram consideradas para o administrador da rede. A hierarquização dos símbolos segue uma convenção que é perfeitamente entendida por quem tem de administrar a rede. No entanto, para o administrador dos ambientes Windows, a hierarquia e os elementos que devem constar

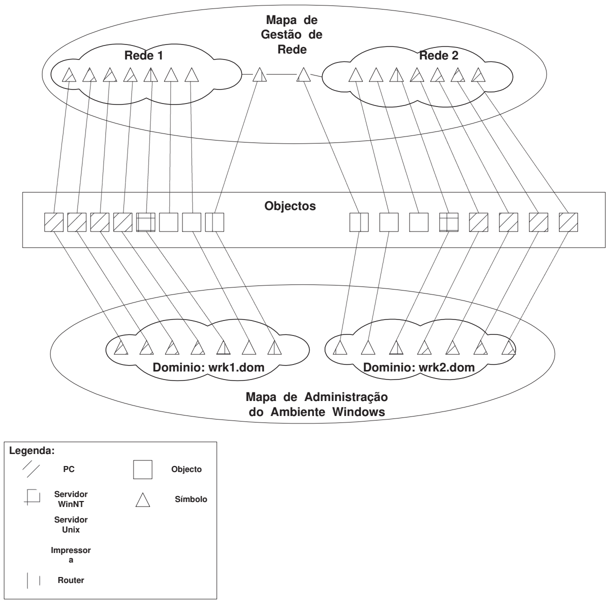

Figura 6.3: Concretização da representação da rede em HP OpenView

da sua vista da rede já deverá ser um pouco diferente. Para este administrador é pouco relevante uma hierarquização baseada nos segmentos de rede IP e fará mais sentido uma organização da rede baseada em domínios NT. No HPOV, a constituição de novas vistas passa pela criação de sub-mapas e de novos símbolos que deverão ser associados aos objectos previamente criados. A Figura 6.3 apresenta de uma forma esquemática a vista do administrador de rede e a vista do administrador dos ambientes Windows.

Obviamente estas duas vistas não esgotam as possíveis representações necessárias à gestão desta infra-estrutura. Uma representação da rede por edifícios e dentro destes por pisos, poderá resultar numa vista interessante para o administrador de rede. A agregação dos computadores pessoais de acordo com os respectivos sistemas operativos ou conforme os recursos aplicacionais utilizados poderá responder a necessidades expressas pelo administrador do ambiente Windows.

A caracterização dos elementos da rede é efectuada através da definição de atributos ao nível dos objectos do HPOV que como vimos anteriormente (Secção 3.2.3) tomam o nome de campos (Fields) e cuja criação é efectuada em ficheiros de registo específicos

```
Field "Localizacao" {
        Type    StringType;
        Flags   locate, general;
}
Field "Departamento" {
        Type    StringType;
        Flags   locate, general;
}
Field "Telefone" {
        Type    StringType;
        Flags   locate, general;
}
Field "Utilizador" {
        Type    StringType;
        Flags   locate, general;
}
Field "Tomada" {
        Type    StringType;
        Flags   locate, general;
}
```

Figura 6.4: Ficheiro de registo de campos

do HPOV (registration files). Estes ficheiros, que possuem uma gramática própria, são interpretados pelo sistema que integra os novos campos em todos os objectos do repositório. A Figura 6.4 apresenta um extracto do ficheiro de registo que foi criado para concretizar os requisitos de informação que foram especificados na secção 6.1. A actualização dos campos pode ser efectuada ao nível da interface gráfica ou através de programas que actuem directamente sobre o serviço `ovwdb`. A Figura 6.5 mostra a visualização dos campos associados a um objecto. Estas caracteristicas levantaram dois problemas na concretização da especificação de teste:

- Um campo ao ser definido, é passível de ser actualizado e mantido em qualquer objecto existente no repositório de objectos do HPOV. No nosso exemplo, o campo “domínio NT” é passível de ser actualizado num objecto associado a um encaminhador, tal como o campo endereço WAN pode ser actualizado num objecto associado a um computador pessoal.

- Todos os símbolos associados a um objecto dão acesso a todos os campos do objecto, não existindo forma de apresentar os campos de uma forma sensível ao sub-

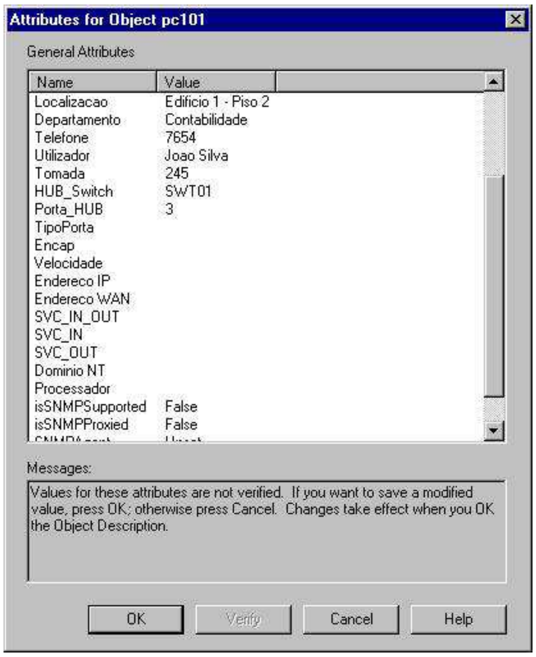

Figura 6.5: Visualização dos campos de um objecto no HP OpenView

mapa onde se insere o símbolo. O administrador dos ambientes Windows ao ter presente no seu sub-mapa os encaminhadores por omissão, tem também acesso a todos os pormenores associados à configuração destes equipamentos (interfaces WAN, protocolos de ligação adoptados, endereços WAN, etc).

O HPOV encontra-se fortemente vocacionada para a adopção do SNMP como protocolo de gestão de rede. Esta característica manifesta-se claramente nas funcionalidades disponíveis na plataforma para obtenção de informação associada aos elementos da rede representados nos seus sub-mapas:

- SNMP MIB Browser permitindo ao utilizador a navegação na árvore de objectos da MIB associada ao elemento da rede.

- Agregação de pedidos SNMP (estatísticas da interface, tráfego da interface, endereços das interfaces, etc). Cada agregação materializa-se no HPOV numa opção de menu, cuja utilização é sensível às características dos objectos seleccionados (caso

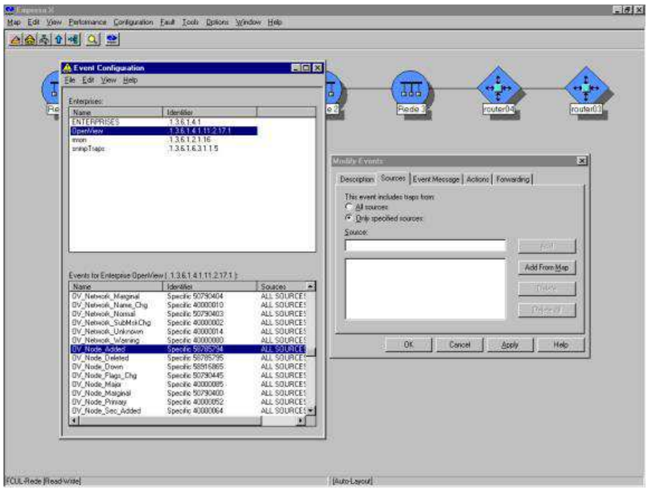

Figura 6.6: Interface de definição do tratamento de eventos no HPOV

se encontre seleccionado um objecto que não disponha de um agente SNMP, não fará sentido a disponibilização de funções associadas a pedidos de informação via SNMP).

- Agregação de pedidos SNMP definidos pelo utilizador e que são integráveis como opções de menu da interface gráfica do HPOV.

O HPOV, distingue a informação que caracteriza os objectos (o conteúdo dos seus campos) daquela que caracteriza o respectivo elemento da rede (normalmente obtida através de SNMP). Embora esta plataforma de gestão disponha de um mecanismo de representação da rede, não permite ao utilizador abstrair-se dos protocolos de gestão de rede, uma vez que as características e estado do elemento da rede têm de ser obtidos explicitamente através de funções criadas para esse efeito. Ou seja, o HPOV não dispõe de nenhum mecanismo que permita relacionar de uma forma automática, as características do elemento de rede, com os campos do objecto que o representa.

O HPOV dispõe de um serviço - `pmd` (HP OpenView Postmaster service) - que é responsável pelo intercâmbio de informação entre todas as aplicações associadas à plataforma. Este serviço tem um papel fundamental no processo de tratamento de eventos, sejam eles originários dos elementos da rede ou das aplicações. As notificações SNMP enviadas pelos elementos da rede são recebidas por um serviço designado por `ovtrapd` que as encaminha para o serviço `pmd`. Este último pode registar a ocorrência em ficheiro (`log`), executar acções ou reencaminhar a notificação para outras aplicações. Os eventos internos às aplicações também são materializadas em notificações que são recebidas pelo serviço `pmd`. O comportamento do sistema perante a ocorrência de um evento pode ser definido através do desenvolvimento de aplicações que utilizem a biblioteca de funções do HPOV, ou através da função "Event Configuration" que é disponibilizada na interface gráfica (Figura 6.6). O HPOV permite que o tratamento de um evento seja sensível ao objecto, possibilitando uma especificação mais fina das características e comportamento dos elementos da rede. Nos sub-mapas e respectivos símbolos, o tratamento de eventos resume-se à alteração da cor do ícone. Esta plataforma não se encontra vocacionada para permitir um tratamento selectivo de eventos adaptado às necessidades de cada vista.

6.2.2 Concretização em Tnm::Map

Devido às características do `Tnm:map`, a representação da rede em estudo correspondeu à criação dos procedimentos que definem os elementos da rede, as funções de tratamento de eventos e de algumas tarefas que possibilitam a representação do estado e comportamento da rede. Na Figura 6.7 é apresentado um extracto desse procedimento relativo à criação de uma das redes.

Como vimos anteriormente, com o `Tnm::map`, a descrição da rede e dos seus elementos é efectuada através de items (`node`, `port`, `network`, `link`) que podem ser agrupados e hierarquizados em grupos (outro tipo de item). A representação das diferentes vistas (rede, ambiente Microsoft Windows, servidores Unix) pode ser efectuada sobre dois modelos: em mapas diferentes ou num mesmo mapa em grupos distintos. A

```
# Criação rede1
set net1 [$maparede create network -name rede1 -tags {rede rede1}]
set no1(1) [$maparede create node -name pc101 -address 192.168.1.101 -tags {pc rede1}]
set no1(2) [$maparede create node -name pc102 -address 192.168.1.102 -tags {pc rede1}]
set no1(3) [$maparede create node -name pc103 -address 192.168.1.103 -tags {pc rede1}]
set no1(4) [$maparede create node -name pc104 -address 192.168.1.104 -tags {pc rede1}]
set no1(5) [$maparede create node -name ux01 -address 192.168.1.10 -tags {Unix server rede1}]
set no1(6) [$maparede create node -name nt01 -address 192.168.1.11 -tags {NT server rede1}]
set no1(7) [$maparede create node -name pr101 -address 192.168.1.20 -tags {printer rede1}]
set no1(8) [$maparede create node -name router01 -address 192.168.1.1 -tags {router rede1}]
```

Figura 6.7: Representação de uma rede em Tnm::map

constituição de vistas é condicionada por algumas das regras que o `Tnm:map` impõe na interligação de items: i) Um item só pode pertencer a um mapa; ii) Um item só pode ter um grupo-pai; iii) A ligação entre nós é efectuada através de elos que se ligam aos primeiros por portas.

Disto resulta que um representante de um elemento da rede não pode ser incluído em mais do que um mapa ou mais do que um grupo. A constituição de mais do que uma vista, seja ela materializada em mapas ou em grupos, tem como consequência a existência de representações redundantes dos elementos da rede.

A Figura 6.8, apresenta de uma forma esquemática, a concretização inicial das vistas do administrador da rede e do administrador dos ambientes Windows, onde se optou pela associação de cada vista a um mapa distinto. O facto dos elementos da rede serem representados por items distintos em cada um dos mapas, teve consequências na definição das características dos elementos da rede:

- Os atributos comuns do elemento (localização, departamento, telefone, etc.) foram associados, de uma forma redundante, aos items que o representavam nos diferentes mapas. Como se poderá verificar na Figura 6.9, foi constituida uma arvore na estrutura de nomes dos atributos que distingue a informação comum da específica.

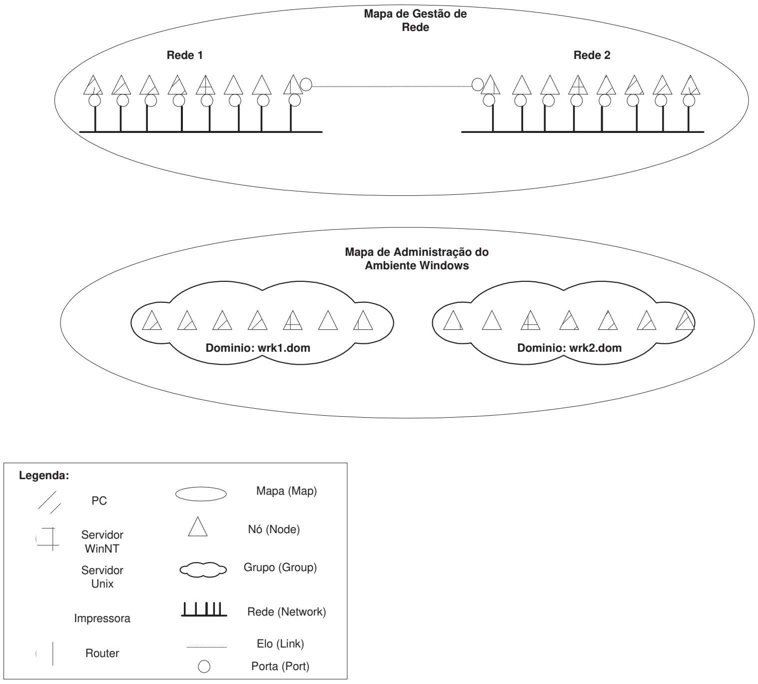

Figura 6.8: Concretização da representação da rede em Tnm::Map

- Os atributos específicos foram distribuídos pelos items tendo em conta o contexto associado à vista.

Para eliminar o problema da incoerência de dados entre items associados a um mesmo elemento da rede, foi necessário desenvolver um procedimento que periodicamente actualiza os atributos comuns alterados. A estratégia de actualização periódica teve de ser adoptada devido ao facto do `Tnm::map` não disponibilizar um mecanismo de tratamento de eventos internos. A existência desse mecanismo permitiria que a ocorrência de uma alteração de um atributo de um item fosse associada a um procedimento responsável pela actualização dos restantes items associados a um mesmo elemento da rede. A concretização do procedimento actualização de atributos comuns obrigou ao estabelecimento de algumas regras:

- Foi necessário encontrar um identificador comum para todos os items associados a um mesmo elemento da rede. Convencionou-se que o nome do items tem de ser

```
# PC101 no mapa de rede
$no1(1) attribute :Comum:Localizacao:Morada.Valor "Edificio1"
$no1(1) attribute :Comum:Localizacao:Piso.Valor "Piso 2"
$no1(1) attribute :Comum:Localizacao:Telefone.Valor "2456"
$no1(1) attribute :Comum:Organizacao:Departamento.Valor "Contabilidade"
$no1(1) attribute :Comum:Organizacao:Sector.Valor "Facturacao"
$no1(1) attribute :Comum:Organizacao:Utilizador.Valor "Joao Silva"
$no1(1) attribute :Rede:Conexao:Tomada.Valor "245"
$no1(1) attribute :Rede:Conexao:Switch.Valor "swt01"
$no1(1) attribute :Rede:Conexao:Porta.Valor "2"
$no1(1) attribute :Rede:SNMP:Comunidade.Valor "public"

# PC101 no mapa de ambiente windows
$win1(1) attribute :Comum:Localizacao:Morada.Valor "Edificio1"
$win1(1) attribute :Comum:Localizacao:Piso.Valor "Piso 2"
$win1(1) attribute :Comum:Localizacao:Telefone.Valor "2456"
$win1(1) attribute :Comum:Organizacao:Departamento.Valor "Contabilidade"
$win1(1) attribute :Comum:Organizacao:Sector.Valor "Facturacao"
$win1(1) attribute :Comum:Organizacao:Utilizador.Valor "Joao Silva"
$win1(1) attribute :NT:Dominio.Valor "TEST.DOM"
$win1(1) attribute :Hardware:Processador.Valor "PII"
$win1(1) attribute :Hardware:Relogio.Valor "166"
$win1(1) attribute :Hardware:RAM.Valor "64"
$win1(1) attribute :Hardware:Disco.Valor "1.2"
```

Figura 6.9: Ficheiro de registo de campos (continuação)

idêntico.

- Por cada atributo comum foi criado um atributo adicional que mantém um número de versão da última alteração do atributo (Exemplo de um atributo de versão "`:Comum:Localizacao:Morada:Versao`"). Os atributos de versão são iniciados com o valor 0 no momento da criação do item e incrementados sempre que o atributo for alterado. Foi convencionado que todos os procedimentos que manipulam os atributos dos items devem actualizar as respectivas versões dos atributos.

- O procedimento de actualização corre periodicamente actualizando os atributos que detêm versões de actualização mais antigas. Foi utilizada a extensão `Tnm::job` para suporte a esta execução periódica [58].

Esta concretização revela as dificuldades encontradas na adopção do `Tnm::map` como ferramenta de suporte a múltiplas representações da rede: o utilizador tem de se preocupar com questões que deveriam ser endereçadas pela ferramenta: coerência dos dados, mecanismos de exclusão mútua (no acesso ao atributo versão), etc.

Desenhou-se ainda um segundo modelo de representação da rede (Figura 6.10) que evita a duplicação dos atributos. Foi constituindo um mapa, designado por mapa repositório, que tem como função manter a representação dos elementos da rede. Os seus itens mantêm os atributos que caracterizam os respectivos elementos da rede. As vistas materializam-se em mapas distintos, designados por mapas especializados, cujos itens mantêm referências com os itens do mapa repositório (cada item de um mapa especializado mantém o testemunho do respectivo item no mapa repositório). Um item de um mapa especializado mantém os atributos específicos ao contexto da vista; um item do mapa repositório mantém os atributos comuns que caracterizam o elemento da rede.

A consulta dos atributos comuns passa agora a ser efectuada em dois passos: o primeiro obtém-se o identificador do item no mapa especializado e posteriormente acede-se aos atributos comuns no mapa repositório. Neste novo modelo, foi necessário convencionar ainda um conjunto de regras associadas à inserção e remoção dos items (Um exemplo: a remoção de um item do mapa repositório só pode ocorrer quando não existir nunhum item a referencia-lo nos mapas especializados).

Este segundo modelo vem demonstrar que se consegue utilizar o `Tnm::map`, na criação de um sistema de vistas que responde a algumas das características que motivaram o desenvolvimento do `Tnm::repo`. Mas isso é conseguido à custa de um esforço adicional de programação e da manutenção de algumas convenções no desenvolvimento da solução. A coerência da representação da rede encontra-se dependente dessas convenções.

O principal obstáculo com que nos deparamos reflecte-se na impossibilidade de um item poder ser associado a mais do que um mapa ou a mais do que um grupo. Adicionalmente, o facto da ligação entre nós (`nodes`) ser obrigatoriamente efectuada através de portas (`ports`), embora possa ser considerada coerente com as representações de

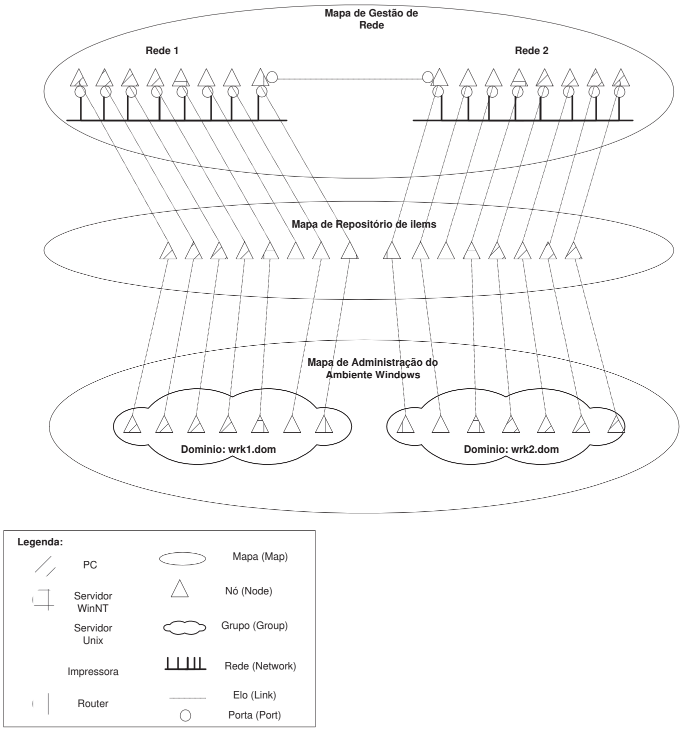

Figura 6.10: Segunda concretização da representação da rede em Tnm::Map

redes de computadores mais comuns, poderá não fazer sentido noutros contextos de representação (processos, componentes, etc).

Para proporcionar o acesso à informação dos agentes, o Scotty dispõe de um conjunto de extensões que disponibilizam interfaces para vários protocolos de gestão, incluindo os protocolos ICMP e SNMP. Com o objectivo de avaliar o grau de abstracção que o `Tnm::map`/Scotty em relação aos protocolos de gestão de rede efectuaram-se dois testes: criação dos procedimentos necessários à verificação da conectividade entre estação de gestão e os elementos da rede, e a actuação sobre a informação contida no agente. A Figura 6.11 descreve o fluxo de funcionamento do sistema de fiscalização.

Os passos 1,2 e 3 foram concretizados num procedimento TCL, cuja execução periódica é controlada através da extensão `Tnm::job`. Este procedimento cria em cada execução uma lista dos nós que suportam o protocolo ICMP (os itens contêm na opção `tag` um identificador designado por "icmp") e o último estado conhecido, que serve

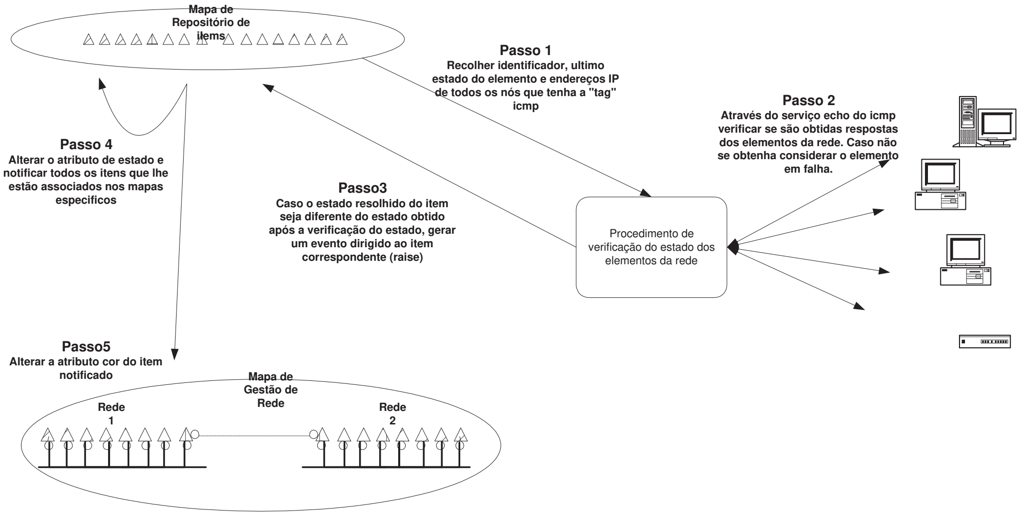

Figura 6.11: Tratamento de eventos utilizando o Tnm::map

de base à verificação da conectividade através de *icmp*. Os nós em que se verificou a alteração de estado são alvo de uma notificação (`raise`).

O passo 4 foi concretizado com um procedimento TCL que foi integrado no mecanismo de tratamento de eventos disponibilizado pelo `Tnm::map` (`bind`). Este procedimento tem como objectivo alterar o atributo de estado do item afectado pelo evento e activar a notificação dos items associados nos mapas específicos. Este procedimento poderia ser definido nos items ou no mapa, uma vez que o `Tnm::map` propaga o tratamento do evento pela hierarquia. Torna-se claro que associar tratamento de eventos ao mapa possibilita o tratamento de eventos generalista. O tratamento de eventos ao nível dos items permite a sua especialização. Uma vez que no nosso caso o tratamento é sempre idêntico, optamos por o associar ao próprio mapa.

O procedimento associado ao passo 5 é conceptualmente idêntico ao que foi efectuado para o passo 4: a notificação gerada no passo 4 tem como consequência a alteração do atributo cor no item do mapa especializado.

A actuação sobre os elementos da rede, foi testada através da criação de um procedimento de visualização e alteração da instância da MIB associada à localização do

```
proc getLocation { ipaddr community } {
        set sn [Tnm::snmp generator -address $ipaddr -community $community]
        set res [$sn get sysLocation.0]
        set res [lindex [lindex $res 0] 2]
        $sn destroy
        return $res
}
proc setLocation { ipaddr community newvalue} {
        set sn [Tnm::snmp generator -address $ipaddr -community $community]
        $sn set [list [list sysLocation.0 $newValue]]
        $sn destroy
}
```

Figura 6.12: Exemplo do acesso à informação do elemento da rede através de SNMP

equipamento (internet.mgmt.mib-2.system.sysLocation) conforme é apresentado na Figura 6.12. Como se poderá observar, no `Tnm::map`, a actuação sobre o elemento da rede é independente da sua representação, não sendo possivel ao utilizador abstrair-se dos protocolos de gestão de rede.

Dos testes de avaliação efectuados retiraram-se algumas conclusões:

- A interligação entre os elementos e a sua representação é efectuada obrigatóriamente por procedimentos externos à representação. Isto verifica-se nos dois testes efectuados uma vez que em qualquer dos casos existe uma clara distinção entre quem detém o papel de representação da rede (mapa) e quem actua sobre a rede (procedimentos de fiscalização e de actuação sobre o elemento da rede).

- O `Tnm::map` não tem qualquer mecanismo que permita relacionar os atributos do item com a informação contida no respectivo agente. Esta limitação tem como consequência a impossibilidade do utilizador se abstrair da interacção explicita com a rede: apesar de dispor da sua representação é obrigado a recorrer a procedimentos externos para poder interagir sobre ela.

- O facto do `Tnm::map` não permitir a integração de vistas que se reflectiu na criação de uma representação da rede baseada em vários mapas distintos (repositório e especializados), tem consequências directas sobre o tratamento de eventos, nomeadamente na obrigatoriedade de criação de procedimentos externos à extensão para a propagação de eventos de um mapa para outro.

- No tratamento de eventos, verifica-se a possibilidade de ocorrência de uma considerável redundância na associação de procedimentos ao tratamento de eventos (`binding`) uma vez que esta associação é efectuada com o próprio procedimento.

Iremos verificar nos testes de avaliação do `Tnm::repo` como estes problemas foram endereçados.

6.2.3 Concretização em Tnm::repo

Analogamente à utilização da extensão `Tnm::map`, a caracterização da rede através da extensão `Tnm::repo` correspondeu à criação de um procedimento TCL. Este procedimento, iniciado pela criação de um repositório, demonstra a existência de duas fases distintas na definição e caracterização da rede: definição dos elementos da rede e definição das vistas.

Na definição dos elementos da rede recorreu-se à facilidade de clonagem que o `Tnm::repo` disponibiliza, a partir de descrições genéricas de elementos de rede comuns (computadores pessoais, encaminhadores, servidores NT, servidores Unix, impressoras). A Figura 6.13 apresenta um extracto da criação dos elemento-nuclear genéricos e da actividade de clonagem.

Criados os dispositivos, seguiu-se a criação das diferentes vistas constantes da especificação referida na secção 6.1. A Figura 6.14 descreve de uma forma esquemática o modelo que foi seguido na representação da rede. A criação das vistas pode ser efectuada de duas formas distintas: criação de vistas com filtros e com a criação manual dos respectivos embaixadores. A Figura 6.15 apresenta um extracto do procedimen-

```
# Criar Repositorio
namespace import Tnm::*
set rp [Tnm::repo create -name empresax]

# Criacao do elemento-nuclear base
set pcbase [$rp create celem -subtype pc -name pcbase]

$pcbase attribute set :Comum:Caracteristicas:SUPIP "Sim"
$pcbase attribute set :Comum:Caracteristicas:EWINDOWS "Sim"
$pcbase attribute set :Comum:Localizacao:Morada ""
$pcbase attribute set :Comum:Localizacao:Piso ""
$pcbase attribute set :Comum:Localizacao:Telefone ""
$pcbase attribute set :Comum:Organizacao:Departamento ""
$pcbase attribute set :Comum:Organizacao:Utilizador ""
$pcbase attribute set :Rede:Conexao:Tomada ""
$pcbase attribute set :Rede:Conexao:Switch ""
$pcbase attribute set :Rede:Conexao:Porta ""
$pcbase attribute set :Rede:SNMP:Comunidade ""
$pcbase attribute set :Estado:ICMP:Estado "Unkn"
$pcbase attribute set :Estado:ICMP:DtUltEstado ""
$pcbase attribute set :Ip:Addr ""

set routerbase [$rp create celem -subtype router -name routerbase]
$routerbase attribute set :Comum:Caracteristicas:SUPIP "Sim"
$routerbase attribute set :Comum:Localizacao:Morada ""
$routerbase attribute set :Comum:Localizacao:Piso ""

set celm(1) [$pcbase clone -name pc101 ]
set celm(2) [$pcbase clone -name pc102 ]

$celm(1) attribute set :Rede:Conexao:Tomada 32
```

Figura 6.13: Ficheiro de registo de campos (continuação)

to desenvolvido onde figura a constituição das vistas com filtros. Esta facilidade do `Tnm::repo`, reduziu a tarefa de criação da vista e a respectiva inclusão dos embaixadores, a um único comando. Uma vez que o filtro definido na vista só se aplica aos embaixadores, foi possível adicionar às vistas os grupos e elos necessários à caracterização dos diferentes segmentos de rede ou domínios NT. A definição manual das vistas implica que seja o utilizador a incluir em cada vista os embaixadores dos elementos que pretende ver representados, tal como se ilustra na Figura 6.16.

Em qualquer dos dois tipos de vistas criados, foi possível caracterizar os atributos associados aos elementos da rede que deveriam ser visíveis em cada uma das vistas, mediante a inclusão nestas de restrições (`constraints`). Esta facilidade permitiu moldar as características dos embaixadores ao contexto da vista onde se inserem, algo que não

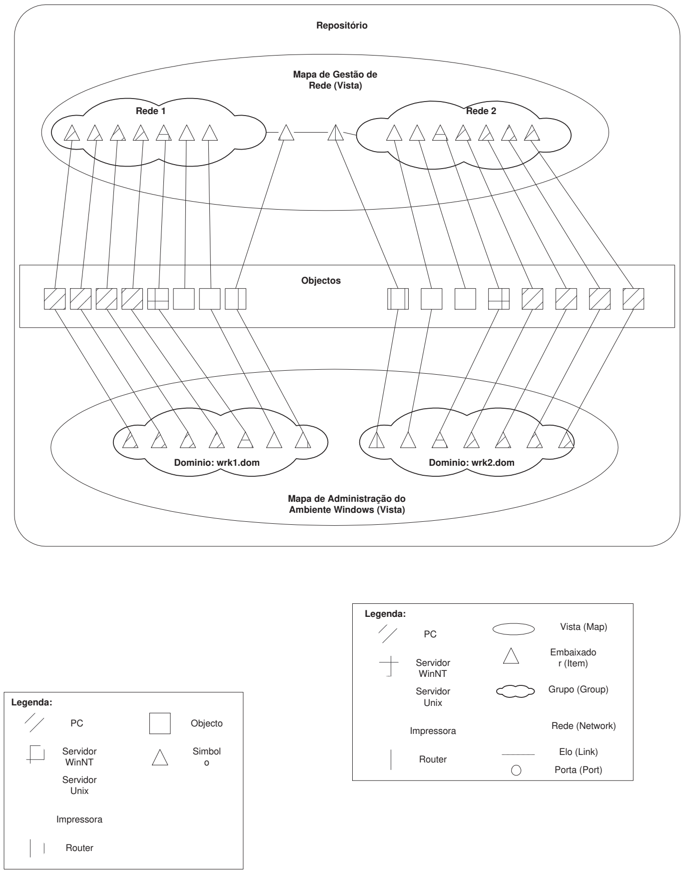

Figura 6.14: Concretização da representação da rede em Tnm::repo

```
set filtro [list :Comum:Caracteristicas:SUPIP == Sim ]
set constr [list [list permit :Comum:] [list permit :Rede:] [list permit :Events:Icmp]]
set nview1 [$rp create map -name allip -filter $filtro]
$nview1 configure -constraints $constr

set filtro [list :Comum:Caracteristicas:EWINDOWS == Sim]
set constr [list [list permit :Comum:] [list permit :Win:]]
set nview2 [$rp create map -name allwindows -filter $filtro]
$nview1 configure -constraints $constr
```

Figura 6.15: Vistas concretizadas em Tnm::repo através de filtros

```
set constr [list [list permit :Comum:] [list permit :Rede:]]
set nview2 [$rp create map -name maparede -constraint $constr
set nelm2(1) [$nview create elem $celm(1) -name [$celm(1) cget -name]]
set nelm2(2) [$nview create elem $celm(2) -name [$celm(2) cget -name]]
set nelm2(3) [$nview create elem $celm(3) -name [$celm(3) cget -name]]
set nelm2(4) [$nview create elem $celm(4) -name [$celm(4) cget -name]]
set nelm2(5) [$nview create elem $celm(5) -name [$celm(5) cget -name]]
set nelm2(6) [$nview create elem $celm(6) -name [$celm(6) cget -name]]
set nelm2(7) [$nview create elem $celm(7) -name [$celm(7) cget -name]]
set nelm2(8) [$nview create elem $celm(8) -name [$celm(8) cget -name]]
set nelm2(9) [$nview create elem $celm(9) -name [$celm(9) cget -name]]
set nelm2(10) [$nview create elem $celm(10) -name [$celm(10) cget -name]]
set nelm2(11) [$nview create elem $celm(11) -name [$celm(11) cget -name]]
set nelm2(12) [$nview create elem $celm(12) -name [$celm(12) cget -name]]
set nelm2(13) [$nview create elem $celm(13) -name [$celm(13) cget -name]]
set nelm2(14) [$nview create elem $celm(14) -name [$celm(14) cget -name]]

set xmemb [$nelm2(1) $nelm2(2) $nelm2(3) $nelm2(4)]
set ngrp1 [$nview create group -name rede1 -members $xmemb]
set xmemb [$nelm2(11) $nelm2(12) $nelm2(13) $nelm2(14)]
set ngrp2 [$nview create group -name rede2 -members $xmemb]
```

Figura 6.16: Criação explicita de uma vista em Tnm::repo

foi possível obter nas restantes soluções em estudo.

O `Tnm::repo` permitiu a criação de vistas adaptadas às necessidades de cada administrador sem que isso implicasse a existência de informação redundante. Esta característica foi conseguida graças à distinção entre dispositivos ("core-elements"), únicos detentores da informação que caracteriza cada elemento da rede, dos embaixadores ("elements") que associados aos primeiros, encontram-se mais vocacionados para representar as múltiplas relações em que cada elemento de rede pode estar envolvido. Este modelo não é obviamente inovador, sendo também suportado pelo HP OpenView.

A avaliação do grau de abstracção dos protocolos de gestão de rede revelou resultados positivos com a adopção de uma classe especial de atributos que é disponibilizada pela extensão: métodos. Assim, a verificação da conectividade do sistema de gestão com os diferentes elementos da rede resumiu-se à inclusão no repositório de um procedimento. Este foi posteriormente associado a cada embaixador tal como é apresentado na Figura 6.17. A fiscalização periódica foi concretizada através da definição da opção "-expire", tornando desnecessária a utilização de procedimentos externos ao repositório

```
set scr1 $rp create script -name ElemStat {
    set addr [%H cget -address]
    set res [icmp echo $addr]
    set xstat [lindex $res 1]
    if { [string length $xstat] == 0 } {
        set xstat "Down"
    } else {
        set xstat "Up"
    }
    set laststate [%H attribute get :Estado:ICMP:Estado]
    if { %L == $xstat } return
    set hora [clock format [clock seconds] -format "%y%m%d%H%M%S"]
    %H attribute set :Estado:ICMP:DtUltEstado $hora
    set ElemStat $xstat
}

# script de activacao de verificacao do estado dos elementos
set elements [$nview1 find -type elem]
foreach xelem $elements {
    set supicmp [$xelem attribute get :Comum:Caracteristicas:SUPIP]
    if { $supicmp == "Sim" } {
       $xelem attribute set :Estado:ICMP:Estado -script $scr1 -expire 180 -poll yes
       $xelem attribute set :Estado:ICMP:DtUltEstado 0
    }
}
```

Figura 6.17: Exemplo da utilização do Tnm::repo em processos de fiscalização

parta concretizar as actividades de fiscalização.

Esta abstracção entre a representação da rede e os protocolos de gestão da rede encontra-se também patente na forma de manipulação da informação dos agentes. Como poderá ser observado na Figura 6.18, na óptica do utilizador da vista, a manutenção da informação constante dos agentes pode resumir-se à manipulação dos atributos dos respectivos Embaixadores/Dispositivos.

No tratamento de eventos, O `Tnm::repo` dispõe de três fluxos distintos de propagação de um evento, ascendente, descendente e lateral.

A utilização da propagação ascendente dos eventos possibilita aos eventos que ocorram num determinado elemento da vista serem tratados nas entidades hierarquicamente superiores a este, algo que pode ser utilizado para o tratamento genérico de eventos (o evento em vez de ser tratado ao de um embaixador, é tratado ao nível do grupo a que pertence ou ao nível da vista) ou para fazer reflectir o evento na hierarquia de entida-

```
set scr23 $rp create script -name snmpLocation {
    switch %O {
        get {
            set ipaddr [%H attribute get :Ip:Addr]
            set community [%H attribute get :Rede:Snmp:Comunidade]
            set sn [Tnm::snmp generator -address $ipaddr -community $community]
            set res [$sn get sysLocation.0]
            set res [lindex [lindex $res 0] 2]
            $sn destroy
            set snmpLocation $res
        }
        set {
            set ipaddr [%H attribute get :Ip:Addr]
            set community [%H attribute get :Rede:Snmp:Comunidade]
            set sn [Tnm::snmp generator -address $ipaddr -community $community]
            $sn set [list [list sysLocation.0 $snmpLocation]]
            $sn destroy
        }
}

# script de activacao do acesso estado dos elementos
set celements [$rp find -type celem]
foreach xcelem $celements {
    set supicmp [$xcelem attribute get :Comum:Caracteristicas:SUPIP]
    if { $supicmp == "Sim" } {
       $xcelem attribute set :Rede:Snmp:SysLoc -script $scr23 -expire 0 -poll no
    }
}

# Acesso ao atributo Localizacao numa vista
set loc [$nelm2(3) attribute get :Rede:Snmp:SysLoc]
```

Figura 6.18: Exemplo de um acesso à informação do elemento da rede através de SNMP

```
$scr1 -script {
    set addr [%H attribute get :Ip:Addr]
    set res [icmp echo $laddress]
    set xstat [lindex $res 1]
    if { [string length $xstat] == 0 } {
        set xstat "Down"
    } else {
        set xstat "Up"
    }
    set laststate [%H attribute get :Estado:ICMP:Estado]
    if { %S == $xstat } return
    set hora [clock format [clock seconds] -format "%y%m%d%H%M%S"]
    %H attribute set :Estado:ICMP:DtUltEstado $hora
    set ElemStat $xstat
    set evt ":Events:Icmp:"
    append evt $xstat
    %H raise -up $evt
    set ElemStat $xstat
}

set scr2 [$rp create script -name ChangeColor1 -script {
    set cnt [%H attribute get :Estado:NDown]
    if { %P == ":Events:Icmp:Down" } {
        %H configure -color "red"
        incr cnt
        %H attribute get :Estado:NDown $cnt
    } else {
        if { $cnt < 2 } {
           %H configure -color "green"
           %H attribute get :Estado:NDown $cnt
        }
    }
}

$nview2 bind :Events:Icmp:Down -script $scr2 -up
```

Figura 6.19: Exemplo da propagação de eventos no Tnm::repo

des acima do elemento da vista. Como vimos anteriormente, esta facilidade encontra-se também disponível no `Tnm::map` mas da sua utilização resulta a existência de procedimentos redundantes no tratamento de um mesmo tipo de evento em itens distintos do mapa. Este problema foi resolvido no `Tnm::repo` com a possibilidade de definir procedimentos de tratamento de eventos ao nível do repositório, que podem ser referenciados por qualquer entidade para o tratamento de determinado tipo de eventos. A Figura 6.19 apresenta as alterações efectuadas ao procedimento apresentado na Figura 6.17, com o objectivo de fazer reflectir na vista o estado dos elementos da rede.

Os eventos descendentes podem ser utilizados para propagar um evento gerado num determinado elemento de uma vista para o dispositivo respectivo e para os di-

```
set scr5 $rp create script -name ChangeVar1 {
     %H attribute set :Rede:Snmp:Comunidade "privada"
}

set elements [$nview1 find -type elem]
foreach xelem $elements {
    $xelem bind :Events:Snmp:ChangeComunity -script $scr5 -down
}
```

Figura 6.20: Exemplo da aplicação do tratamento de eventos descendentes

ferentes embaixadores associados a este último. Deste modo é possível propagar um evento ocorrido numa vista para as restantes vistas existentes no repositório. De notar que esta propagação pode ser limitada através das restrições à vista, especializando desta forma o tratamento dos eventos ao contexto da vista. Para demonstrar esta capacidade foi desenvolvido um procedimento de tratamento do evento de "disco cheio" que foi associado aos computadores pessoais da vista do administrador do ambiente Windows. A restrição de recepção deste tipo de evento na vista do administrador da rede impediu que este recebesse informação irrelevante no seu contexto.

A propagação descendente dos eventos foi ainda utilizada para enviar ordens de execução de comandos por todos os elementos de uma vista, como se pode observar na Figura 6.20. A execução do comando "`$nview1 raise :Events:Snmp:ChangeComunity -down`" tem como consequência a alteração do atributo `:Rede:Snmp:Comunidade` em todos os elementos da vista.

Ao `Tnm::repo` foi ainda acrescentada a facilidade de tratamento de eventos internos, algo que não se encontra disponível no Tnm::map e que no HPOV só é disponibilizado na biblioteca de desenvolvimento em C. Esta facilidade permite refinar o comportamento das diferentes entidades envolvidas na representação da rede. É assim possível efectuar tratamentos adicionais associados à inserção ou remoção de novos elementos no repositório, à criação e remoção de vistas e procedimentos, à alteração de atributos ou sua remoção, etc.

Da avaliação das funcionalidades do `Tnm:::repo`, resultam sumariamente as seguintes conclusões:

- O Tnm::repo possibilita a criação de representações da rede que podem ser vistas sob múltiplas perspectivas sem que isso seja sinónimo de existência de informação redundante.
- Esta extensão disponibiliza ainda a capacidade de filtrar ao nível da vista, a informação que caracteriza cada elemento da rede, quer ao nível dos atributos quer aos nível dos eventos, moldando, desta forma, cada elemento ao contexto da vista.
- Adicionalmente o `Tnm::repo` disponibiliza mecanismos mais poderosos que as restantes soluções em estudo, para o tratamento de eventos: propagação de eventos ascendentes, descendentes e laterais e tratamento de eventos internos. Foi optimizado o processo de associação de procedimentos TCL ao tratamento de eventos através da capacidade de incluir os procedimentos nos repositórios.
- A nível dos atributos também foram disponibilizadas funcionalidades interessantes, nomeadamente os atributos constantes e os métodos. Os métodos apresentam a vantagem de tornar possível ao utilizador da vista a abstracção dos protocolos de gestão de rede.

Verificou-se no decorrer dos testes um inconveniente previsível: tempo de execução que será apresentado na secção seguinte.

6.3 Avaliação do Desempenho

Nos testes de avaliação do desempenho da extensão `Tnm::repo`, optou-se por considerar somente o `Tnm::map` como termo de comparação, devido aos seguintes factores:

- O `Tnm::repo` pretende ser uma evolução do `Tnm::map` pelo que faz sentido a realização de testes comparativos de desempenho entre as duas extensões.

- Sendo os dois sistemas extensões à linguagem TCL, é simples realizar testes comparativos uma vez que estamos a comparar sistemas que tentam atingir um determinado objectivo (representação da rede) através de meios semelhantes.

- Comparar o `Tnm::repo` com o HP OpenView parece-nos pouco realista já que este ultimo é uma bancada de gestão com uma forte componente gráfica, dificilmente comparável em termos de desempenho com uma extensão a uma linguagem interpretada.

A avaliação pode ser efectuada sobre duas componentes, eficiência na execução de actividades de gestão e eficiência na execução de comandos. A primeira classe de testes está obviamente relacionada com as capacidades que a extensão proporciona ao utilizador na gestão da representação da rede Esta componente, essencialmente funcional já foi tratada na secção anterior. Resta assim a eficiência na execução de comandos, que terá como resultado um comparativo do desempenho entre extensões.

Nesta avaliação consideraram-se os seguintes testes: $i)$ Alteração do conteúdo de um atributo de um elemento de uma vista; $ii)$ Visualização do conteúdo de um atributo de um elemento de uma vista; $iii)$ Ocorrência de um evento ascendente. Foi considerada a seguinte representação de rede:

- Representação através do `Tnm::map`: Um mapa contendo 20 nós, cada um com 15 atributos. O teste relativo à manipulação de atributos incide sobre o 10º atributo do 15º nó criado. O teste de geração de eventos é também efectuado sobre o 15º nó criado.

- Representação através do `Tnm::repo`: Um repositório contendo 20 dispositivos

```
proc test { obj } {
# Funcao onde e inserido o teste a realizar
}
#Variavel objecto contém o testemunho
#do item ou elemento alvo do teste
set flag 1
set counter 0
set clockini [clock seconds]
incr clockini 60
while { $flag } {
  test $obj
  set clockend [clock seconds]
  if { $clockini < $clockend } {
        set flag 0
        continue
  }
  incr counter
}
puts "Iteracoes por minuto = $counter"
```

Figura 6.21: Procedimento base para a realização de testes de desempenho do Tnm::repo

e três vistas. Cada dispositivo possui 15 atributos. Cada vista tem associados 20 embaixadores. Cada vista tem duas regras de restrição que inibem o acesso aos atributos 4 e 7. Os testes relativos a manipulação de atributos incide sobre o 10º atributo do 15º embaixador. O teste de geração de eventos é também efectuado sobre o 15º embaixador.

O procedimento que serviu de base à realização dos testes encontra-se descrito na Figura 6.21. O objectivo de cada teste é obter o numero de execuções por minuto por forma a conseguir-se medir a eficiência entre as duas extensões. Na realização destes testes utilizou-se um computador pessoal com um processador Pentium a 133 Mhz, com 48 Mb de memória correndo o sistema operativo Linux (Red Hat 5.2). Dos testes efectuados, aplicando os comandos apresentados na Tabela 6.1 aos dois sistemas, obtiveram-se os resultados apresentados na Tabela 6.2. Como se poderá constatar verifica-se uma perca de eficiência acima dos 41%.

Efectou-se outro teste onde se alteraram as condições de teste do `Tnm::repo` que passou a ser um repositório contendo 20 dispositivos, cada um com 15 atributos. Os testes relativos a manipulação de atributos incidem sobre o 10º atributo do 15º dispo-

Instrução Tmn::map	**Instrução Tmn::Repo**
Alteração de Atributo:	
`$obj attribute ::Test:Attr15 "teste"`	`$obj attribute set ::Test:Attr15 "teste"`
Valor do Atributo	
`$obj attribute ::Test:Attr15`	`$obj attribute get ::Test:Attr15 "teste"`
Ocorrência de eventos	
`$obj raise ::Evttst:evt1`	`$obj raise ::Evttst:evt1 -up`

Tabela 6.1: Comandos usados nos testes de desempenho

Teste	N. Iterações (`Tnm:Map`)	N. Iterações (`Tnm::repo`)
Alteração de Atributo	327646	464678
Valor do Atributo	404699	573863
Ocorrência de eventos	181839	257847

Tabela 6.2: Desempenho comparativo entre `Tnm:Map` e `Tnm::repo`

sitivo. O teste de geração de eventos é também efectuado sobre o 15º elemento-nuclear. Neste novo ambiente de teste, não se verificaram diferenças significativas nos resultados obtidos nas duas extensões o que permite concluir que a perca de eficiência da extensão `Tnm::repo` é causada pela componente de validação das restrições da vista e no facto do acesso aos atributos do elemento do `Tnm::repo` terem mais um passo que o acesso aos atributos de um item no `Tnm::map`: Os atributos do embaixador encontram-se residentes no dispositivo.

Da posterior análise dos procedimentos associados ao tratamento das restrições da vista, verificou-se que era possível a sua optimização. No algoritmo original sempre que se verificava um acesso a um atributo, era efectuada a comparação de cada uma das regras de restrição com o nome do atributo, por comparação de caracteres. Por exemplo: se a restrição tivesse 14 caracteres, a comparação seria efectuada sobre os 14 primeiros caracteres do nome do atributo. Se o nome do atributo tivesse menos de 14 caracteres então não cumpre a regra.

Para tornar mais eficiente o processo de controlo de acessos aos atributos, optou-se por associar uma lista de controlo de acesso a cada atributo dos elementos nucleares. Esta lista possui um elemento booleano por cada vista que indica se o atributo é ou nao visivel nela. Este campo é actualizado sempre que verifica alteração das regras de restrição na vista ou quando o atributo é criado. Esta lista de controlo de acesso é materializada como um vector de digitos binários tornando bastante eficiente o seu acesso. Com esta optimização obtiveram-se 348782 iterações por minuto no acesso a atributos já existentes no elemento da vista, ou seja, a perca de eficiência no acesso a atributos baixou para aproximadamente 6.5%.

Torna-se evidente que apesar desta optimização, devido ao aumento de funcionalidades que são disponibilizadas na nova extensão, o desempenho do `Tnm::repo` é inferior ao `Tnm::map`. Além disso estamos convictos que esta perca de eficiência será largamente compensada pela redução do código TCL que tem de ser executado para que o `Tnm::map` apresentasse as facilidades adicionais que o `Tnm::repo` apresenta.

Sumário

Este capítulo apresentou a validação do modelo proposto. Foram modelados ambientes de gestão sobre uma rede de comunicações, utilizando o nosso protótipo (`Tnm::repo`) e mais dois sistemas de gestão de rede (`Tnm::map` e HP OpenView).

Verificou-se que o nosso modelo aumentava a capacidade de representação da rede em relação aos sistemas alvo da comparação, nomeadamente na manipulação de diferentes vistas associadas aos mesmos elementos da rede, no tratamento dos atributos dos elementos da rede e no tratamento de eventos. O nosso prototipo apresenta um desempenho menor que o `Tnm::map`, derivado do aumento de funcionalidades.

Capítulo 7

Conclusões

Em sistemas de gestão de redes, uma *vista* representa uma perspectiva parcelar do universo de gestão, adequada a uma actividade ou cargo específico. Esta dissertação abordou o problema de fornecer os mecanismos adequados para o desenvolvimento de plataformas de gestão de redes capazes de suportar simultaneamente diversas vistas dos mesmos objectos de gestão.

Para motivar o problema, a dissertação apresenta uma panorâmica sobre as soluções que têm vindo a ser adoptadas ao nível dos protocolos e das plataformas de gestão. Esta panorâmica refere também o modo como nestas plataformas é abordada a relação entre o administrador de rede e os dispositivos físicos ou lógicos que são alvo da gestão.

Desta panorâmica emergem os seguintes factos:

- As funcionalidades relacionadas com a integração de aspectos de cariz mais tecnológico, fundamentalmente associadas aos mecanismos e protocolos de gestão de rede, têm concentrado grande parte do esforço de desenvolvimento na área.

- A evolução dos sistemas computacionais, quer em termos do seu crescimento quer em termos da sua heterogeneidade, criou a necessidade de existência de ferramentas de gestão que por um lado, ofereçam uma perspectiva global de toda a infra-estrutura de sistemas de informação alvo da gestão mas que também permitam por outro, segmentar e especializar em vistas parcelares, possibilitando a gestão especializada de sub-sistemas.

Verificando-se que as propostas existentes na área de gestão de redes apresentam lacunas no suporte a gestão de vistas, a dissertação propõe um modelo para a gestão de vistas, assim como um conjunto de mecanismos de suporte adequados a este modelo.

O modelo proposto agrega aspectos relevantes que já se podiam encontrar em alguns dos sistemas revistos, embora de forma dispersa e não integrada, e introduz algumas características que se julgam inovadoras. Enumeram-se as características mais relevantes do modelo proposto:

- Separação entre as entidades que representam os elementos da rede e as entidades que representam as múltiplas relações em que estes podem estar envolvidos (característica patente na plataforma HP OpenView).

- Modelo de Eventos que permite caracterizar o comportamento de cada elemento da rede com diferentes graus de refinamento (inspirado no sistema `Tnm::map`).

- Mecanismo de definição de políticas de acesso aos objectos (patente no sistema Domains).

- A capacidade das vistas filtrarem a informação dos elementos da rede que é visível ao utilizador da vista

- A expansão do modelo de eventos do `Tnm::map` por forma a permitir que a execução de eventos tenha a capacidade de se propagar de uma forma ascendente, descendente e lateral

- Permitir ao utilizador abstrair-se dos protocolos de gestão de rede e dos necessários processos de dialogo existentes entre o sistema de gestão e os elementos da rede.

De modo a validar a exequibilidade da concretização deste modelo, foi realizado um protótipo sob a forma de uma extensão ao `Tnm::map`, designada por `Tnm::repo`. Ao nível da concretização, e para além das novas funcionalidades referidas, o `Tnm::repo` apresenta alguns melhoramentos adicionais ao `Tnm::map`, como por exemplo a capacidade de clonagem de dispositivos e associação dos procedimentos ao repositório.

O esforço de concretização através de uma extensão Tcl, reforçou a nossa convicção de que as linguagens interpretadas com as características do Tcl apresentam vantagens únicas na modelação e prototipagem de ferramentas de gestão de rede que compensam largamente a perca de desempenho que as caracterizam.

A flexibilidade do modelo e o desempenho do protótipo foram avaliados através da sua aplicação a um cenário "alvo" e da comparação do sistema resultante com o sistema que resultaria da aplicação de outras ferramentas ao mesmo cenário, nomeadamente dos sistemas HP OpenView e `Tnm::map`.

Em termos de perspectivas de melhoramentos futuros ao sistema proposto, apontam-se as seguintes direcções:

- O encapsulamento e herança múltipla, características das linguagens orientadas por objectos, são duas propriedades interessantes quando pretendemos representar a rede e os seus elementos. A utilização deste paradigma na constituição de um sistema de vistas vocacionado para a gestão de rede, é sem duvida alguma, uma expansão interessante ao nosso trabalho.

- No `Tnm::repo` as restrições são aplicadas ao nível da vista. Seria interessante generalizar o uso das restrições aos diferentes níveis hierárquicos da vista.

- O mecanismo de restrição embora eficaz é elementar e passível de ser melhorado.
- Seria igualmente interessante estender o modelo e os respectivos mecanismos de suporte para permitir a distribuição dos repositórios.

Apêndice A

Alguns ponteiros de interesse

Para a escrita desta dissertação foram consultados vários "locais" na Internet que incluem informação de interesse sobre diferentes sistemas e técnicas de gestão de redes. Para conveniência do leitor, estes são listados de seguida.

Web Tutorials `http:// www.distributed-networking.com/default.htm`
Tuturials da University of New Hampshire `http://www.iol.unh.edu/training/`
The Java Based Management Page `http:// joe.lindsay.net/javamgmt.html`
Computer Science Research Paper Search Engine `http://cora.whizbang.com/`
New Network Management Links de David C Blight (University of Manitoba Canada) `http://www.ee.umanitoba.ca/~blight/telecommunications/nm-new.html`

Network Management Archives `http://smurfland.cit.buffalo.edu/NetMan/Archives.html`
SNMP FAQ `ftp://rtfm.mit.edu/pub/usenet/news.answers/snmp-faq`
Network Management Resources `http://tampico.cso.uiuc.edu/˜chris/netmgmt/`
The Simple Web `http://www.simpleweb.org/`
The Simple TimesNetwork Management related white papers, articles and thesis `http://www.simple-times.org/`

Bibliografia

[1] Sinan Si Alhir. *UML in a Nutshell*. O'Reilly, 1 edition, 1998.

[2] N. Anerousis. An architecture for building scalable, web-based management services. Technical report, AT&T Labs Research.

[3] N. Anerousis. An information model for generating computed views of management information. Technical report, AT&T Labs Research.

[4] Computer Associates. http://www.cai.com.

[5] W. Barr. The tina initiative. *IEEE Communications Magazine*, 1993.

[6] U. Black. *Network Management Standards: The OSI, SNMP, & CMOL Protocols*. 1994.

[7] U. Blumenthal and B. Wijnen. *RFC 2574 - User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*, 1999.

[8] A. Carzaniga, G. Picco, and G. Vigna. Designing distributed applications using mobile code paradigms. Technical report, 1997.

[9] J. Case, M. Fedor, M. Schoffstall, and J. Davin. *RFC 1157 - A Simple Network Management Protocol (SNMP)*, 1990.

[10] J. Case, D. Harrington, R. Presuhn, and B. Wijnen. *RFC 2572 - Message Processing and Dispatching for the Simple Network Management Protocol (SNMP).*, 1999.

[11] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1901 - Introduction to Community-based SNMPv2*, 1996.

[12] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1902 - SMI for SNMPv2*, 1996.

[13] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1903 - Textual conventions for SNMPv2*, 1996.

[14] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1904 - Conformance statements for SNMPv2*, 1996.

[15] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1905 - Protocol operations for SNMPv2*, 1996.

[16] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1906 - Transport mappings for SNMPv2*, 1996.

[17] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1907 - MIB for SNMPv2*, 1996.

[18] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1908 - Coexistence between SNMPv1 and SNMPv2*, 1996.

[19] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1909 - An administrative Infrastructure for SNMPv2*, 1996.

[20] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *RFC 1910 - User-based Security Model for SNMPv2*, 1996.

[21] R. Case, R. Mundy, D. Partain, and B. Stewart. *RFC 2570 - Introduction to Version 3 of the Internet-standard Network Management Framework*, 1999.

[22] Castle Rock Computing. http://www.castlerock.com/.

[23] Red Point Computing. http://www.redpt.com/SnmpQL/index.html.

[24] G. Cugola, C. Ghezzi, G.P. Picco, and G. Vigna. Analyzing mobile code languages. Technical report, 1997.

[25] Desktop Management Task Force. *Desktop Management Interface Specification. Version 2.00*, 1996.

[26] Desktop Management Task Force. *Common Information Model (CIM) Specification. Version 1.1*, 1997.

[27] G. Goldszmidt. *Distributed Management by Delegation*. PhD thesis, Columbia University, New York, 1995.

[28] J . Gosling and H. McGilton. The java language environment: A white paper. Technical report, 1995.

[29] R. Gray. A transportable agent system. Technical report, 1995.

[30] Object Management Group. *Object Management Architecture Guide*, 1992.

[31] Object Management Group. *The OMG Object Model v.0.9*, 1992.

[32] Object Management Group. *The Common Object Request Broker Architecture and Specification*, 1995.

[33] Tivoli Systems Inc. http://www.cai.com.

[34] D. Johansen, R. van Renesse, and F. Schneider. An introduction to the tacoma distributed system. Technical report, 1995.

[35] P. Kalyanasundaram, A. Sethi, C. Sherwin, and D. Zhu. A spreadsheet-based scripting environment for snmp. 1997.

[36] D. Levi, P. Meyer, and B. Stewart. *RFC 257 - SNMP Applications*, 1999.

[37] General Magic. Telescript language reference. Technical report, 1995.

[38] F. Matthes, F. Mussig, and J. Schmidt. Persistent polymorphic programming in tycoon: An introduction. Technical report, 1993.

[39] S. Mazumbar. Translation of snmpv2 specification into cobra-idl. Technical report, 1995.

[40] S. Mazumbar and T. Roberts. Translation of gdmo specification into cobra-idl. Technical report, 1995.

[41] M. McLennan. Object-oriented programing with [incr tcl]. Technical report, Lucent Technologies, 1996.

[42] J. Moffett and M. Sloman. User and mechanism views of distributed systems management. *IEE/IOP/BCS Distributed Systems Engineering*, 1993.

[43] International Standards Organization. *Open System Interconnection - Remote Operations (ROSE): Model, Notation and Service Definition*, 1989.

[44] International Standards Organization. *Open System Interconnection - Remote Operations (ROSE): Protocol Specification*, 1989.

[45] International Standards Organization. *Information Tecnology - OSI, Common Management Information Protocol (CMIP)*, 1990.

[46] International Standards Organization. *Information Tecnology - OSI, Common Management Information Service Definition (CMIS)*, 1990.

[47] International Standards Organization. *Information Processings Systems - OSI - Management Information Systems - Service Definition for Association Control Service Element (ACSE)*, 1992.

[48] International Standards Organization. *Information Processing Systems - Open Systems Interconnection - Structure of Management Information - Part 1: Management Information Model*, 1993.

[49] International Standards Organization. *Information Processing Systems - Open Systems Interconnection - Structure of Management Information - Part 2: State Management Function*, 1993.

[50] International Standards Organization. *Information Processing Systems - Open Systems Interconnection - Systems Maagement Overview*, 1993.

[51] J. Ousterhout. Tcl: An embeddable command language. *1990 Winter Usenix Conference Proceedings*, 1989.

[52] Hewlett Packard. http://www.openview.com.

[53] C. Palma and L. Rodrigues. Tnm::repo manual. Technical report, 2000.

[54] M. Rose and D. Cass. *ISO transport services on top of the TCP: Version 3*, 1987.

[55] M. Rose and K. McCloghrie. *RFC1155 - Structure and Identification of Management Information for TCP/IP-based Internets*, 1990.

[56] M. Rose and K. McCloghrie. *MIB for Network Management of TCP/IP-based internets: MIB-II*, 1991.

[57] J. Schönwälder. *Manual do scotty*. http://wwwsnmp.cs.utwente.nl/s̃choenw/scotty/.

[58] J. Schönwälder. *Manual do scotty*. http://wwwsnmp.cs.utwente.nl/s̃choenw/scotty/.

[59] J. Schönwälder and H. Langendörfer. Ined - an application independent network editor. Technical report, Technical University of Braunschweig, Germany, 1993.

[60] J. Schönwälder and H. Langendörfer. Tcl extensions for network management applications. Technical report, Technical University of Braunschweig, Germany, 1995.

[61] M. Sloman and J. Moffett. Domain management for distributed systems. Technical report, 1989.

[62] BMC Software/Cisco/Compaq/Intel/Microsoft. Industry leaders propose web-based enterprise management standards, 1996.

[63] J. Stamos and D. Gifford. Remote evaluation. Technical report, 1990.

[64] Sunsoft. *Java Management API Architecture, Revision A*, 1996.

[65] D. Tennenhouse and D. Wetherall. Toward an active network architecture. *ACM Computer Communication Review, 26*, 1996.

[66] S. Waldbusser. *RFC 1271 - Remote Network Monitoring Management Information Base*, 1991.

[67] S. Waldbusser. *RFC 1757 - Remote Network Monitoring Management Information Base*, 1995.

[68] U. Warrier, L. Besaw, L. LaBarre, and B. Handspicker. *Common Management Information Services and Protocols for the Internet (CMOT and CMIP)*, 1990.

[69] C. Wellens and K. Auerbach. Toward useful management. *The Simple Times*, 1996.

[70] B. Wijnen, D. Harrington, and R. Presuhn. *RFC 2571 - An Architecture for Describing SNMP Management Frameworks*, 1999.

[71] B. Wijnen, R. Presuhn, and K. McCloghrie. *RFC 2575 - View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP).*, 1999.

[72] T. Yemini and S. da Silva. Towards programable networks. *Proc. /th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'96)*, 1996.

[73] W. Yeong. Snmp query language. Technical report, Performance Systems International Inc, 1990.