UMA APLICAÇÃO DE GERÊNCIA DE REDES DE COMPUTADORES BASEADA EM WEB

Antonio Augusto Teixeira Ribeiro Coutinho

Dissertação submetida à Coordenação de Pós-Graduação em Informática da Universidade Federal da Paraíba – Campus II, em cumprimento às exigências parciais para obtenção do Grau de Mestre em Informática.

Área de Concentração: Redes de Computadores

Jacques Philippe Sauvé
(orientador)

Campina Grande, Paraíba, Brasil
Setembro, 2000

Resumo

Neste trabalho, desenvolvemos uma aplicação de gerência com o objetivo de pesquisar como algumas das tecnologias empregadas na Web podem ser utilizadas como alternativa para as técnicas de gerência tradicionais. A solução proposta provê um modo apropriado para resolver o problema da mobilidade de interface apresentado pela abordagem tradicional através de um arquitetura seguindo um modelo em três camadas, independente de plataforma e baseada em componentes de software. Suas características principais incluem uma interface Web, navegação através de mapas da rede, indicações automáticas do estado dos dispositivos e sub-redes através de cores, exibição de gráficos estatísticos sobre o comportamento dos dispositivos, alarmes automáticos e boa capacidade de configuração. A arquitetura é flexível, permitindo uma fácil adição de novas funcionalidades.

Abstract

This dissertation presents a Web-based network management application. The objective is to research how Web technologies can be used as alternatives to traditional management techniques. The proposed solution provides an appropriate way to solve the interface mobility problem exhibited by the traditional approach through a three-tier, cross-platform and component-based application. Its main features include a Web-based interface, network navigation maps, display of graphical statistics, automatic color status indications for devices and sub-networks, full configurability and automatic alarms. The architecture is very flexible and allows to be easily added to the application.

Ao meu querido pai, que faleceu durante a realização deste trabalho. Gostaria que soubesse o quanto menos importante e vitorioso este momento se tornou sem sua presença.

Agradecimentos

A Deus, por toda força e perseverança concedida para esta realização.

A minha querida mãe, Lúcia, por enfrentar todos os fatos e dificuldades que surgiram durante este mestrado, encarando muitas vezes sozinha os problemas, enquanto eu buscava concentração para minhas pesquisas. Sem seu amor, seu apoio, sua serenidade e seu exemplo de coragem não teria conseguido.

Ao Prof. Jacques Philippe Sauvé pela sua estimada orientação, confiança e compreensão diante das dificuldades que enfrentei para concluir este trabalho.

Ao amigo Gustavo Henrique Machado de Arruda, por toda ajuda e dedicação.

As amigas Lívia e Flávinha, por compartilhar os momentos de ansiedade.

Aos alunos Lindolfo Pereira Araújo, Osório Abath Neto, Raquel Vigolvino Lopes, Ayla Débora Dantas de Souza e Rodrigo Rebouças de Almeida pela importante contribuição para este trabalho. Juntos, formamos realmente uma grande equipe.

Por fim, a todos me ajudaram e deixei de citar, minha gratidão.

Conteúdo

Capítulo 1 - Introdução 1

1.1 Objetivos da Dissertação 3

1.2 Escopo e Relevância 3

1.3 Estrutura da Dissertação 3

Capítulo 2 - Gerência de Redes de Computadores 5

2.1 O Que é Gerência de Redes 6

2.2 Áreas Funcionais da Gerência de Redes 7

2.3 Arquitetura Básica de Gerência 7

2.4 O Padrão de Gerência Internet 9

2.5 Aplicações e Plataformas de Gerência 11

2.6 Limitações da Arquitetura de Gerência Tradicional 12

2.6.1 Integração 12

2.6.2 Mobilidade 12

2.6.3 Segurança 12

2.6.4 Custos 13

2.7 Arquiteturas Baseadas em Web 13

2.7.1 Arquitetura em Duas Camadas 14

2.7.2 Arquitetura em Três Camadas 15
2.7.3 Outras Iniciativas 16
Capítulo 3 - Uma Aplicação de Gerência de Rede Baseada em Web: Requisitos e Projeto Arquitetural 18
3.1 Protótipo da Aplicação 18
3.1.1 Requisitos 19
3.1.2 Especificação Funcional 20
3.1.3 Arquitetura 21
3.1.4 Análise do Protótipo 23
3.2 Levantamento dos Requisitos 25
3.2.1 Requisitos Funcionais 26
3.2.2 Requisitos Não-Funcionais 29
3.3 O Projeto Arquitetural 31
3.3.1 Arquitetura baseada em Web 33
3.3.2 Aquisição da Informação de Gerência 34
3.3.3 Tratamento e Visualização da Informação de Gerência 36
3.3.5 Componentes da Interface Web 40
3.3.6 Configuração da Aplicação 43
3.3.7 Síntese por Componente Apresentado 44
Capítulo 4 - Projeto Detalhado e Implementação 47
4.1 Organização e Construção do Projeto 47
4.2 APIs Utilizadas 49
4.2.1 API Infobus 49
4.2.2 API AdventNet SNMP 51
4.2.3 API JConfig 51
4.2.4 API Map 52
4.2.5 API Graphic 52
4.2.6 API JavaMail 52
4.3 Pacotes Desenvolvidos 53
4.3.1 Pacote webmngr 53

4.3.2 Pacote webmngr.config 55
4.3.3 Pacote webmngr.monitor 55
4.3.4 Pacote webmngr.log 57
4.3.5 Pacote webmngr.status 57
4.3.6 Pacote webmngr.event 58
4.3.7 Pacote webmngr.info 58
4.3.8 Pacote webmngr.mail 58
4.3.9 Pacote webmngr.servlet 59
4.3.10 Pacote webmngr.util 59
4.4 Procedimentos de Testes Realizados 59
4.5 Interface da Aplicação 61
Capítulo 5 - Uso Prático da Solução: Gerenciando a rede do POP-Pb 62
5.1 A Rede do POP-Pb 62
5.2 Gerenciando a Rede com a Aplicação 63
Capítulo 6 - Conclusões 71
6.1 Avaliação e Satisfação dos Requisitos 71
6.2 Problemas Enfrentados 74
6.3 Trabalhos Futuros 75
Bibliografia 78
Apêndice A - Configuração XML para os Equipamentos do POP-Pb 82
Apêndice B - Configuração XML para os Mapas da Rede do POP-Pb 96
Apêndice C - Diagramas de Classes Desenvolvidas 100
Apêndice D - Glossário 110

Lista de Figuras

Figura 2.1: Tipos de Dispositivos de uma Rede de Computadores.

Figura 2.2: Arquitetura básica de gerência.

Figura 2.3: Primitivas básicas de comunicação do protocolo SNMP.

Figura 2.4: Aplicações de Gerência de Rede.

Figura 2.5: Arquitetura Em Duas Camadas

Figura 2.6: Arquitetura em Três Camadas

Figura 3.1: Visão geral da rede gerada pelo protótipo

Figura 3.2: Informações obtidas sobre um canal de comunicação.

Figura 3.3: Arquitetura do Protótipo da Aplicação

Figura 3.4: Modelo HTML dos mapas da rede

Figura 3.5: Aplicação de Gerência baseada em Web

Figura 3.6: Núcleo da Aplicação de Gerência

Figura 3.7: Arquitetura geral da solução

Figura 3.8: Exemplo de gráfico estatístico sobre variáveis MIB

Figura 3.9: Exemplo de um Mapa da Rede

Figura 3.10: Esquema dos mapas da rede

Figura 3.11: Exemplo de configuração do equipamento usando XML

Figura 4.1: Diagrama de pacotes

Figura 4.2: Diagrama de colaboração I

Figura 4.3: Diagrama de colaboração II

Figura 4.4: Testando as classes da aplicação com o JUnit

Figura 4.5: Modelo de página JSP

Figura 5.1: Página inicial da aplicação

Figura 5.2: Mapa principal da rede do POP-Pb

Figura 5.3: Mapa detalhado do POP-Pb

Figura 5.4: Mapa Serviço Web Proxy do POP-Pb

Figura 5.5: Mapa mostrando o Switch IBM 8271 do POP-PB

Figura 5.6: Informações sobre a interface do Switch IBM 8271

Figura 5.7: Gráfico sobre o Tempo de Ping do Switch 8271 do POP-PB

Figura 5.8: Lista de equipamentos gerenciados pela aplicação

Figura 5.9 : Lista de alarmes do equipamento do Switch IMB 8271

Figura 6.1 : Arquitetura com Servidor de Traps SNMP

Capítulo 1

Introdução

No início, as redes de computadores foram concebidas simplesmente como meio de compartilhar dispositivos periféricos caros como impressoras, discos e modems de alta velocidade. Porém, as redes corporativas foram ficando cada vez maiores e integradas às organizações, com muitas interligações com outras redes, a exemplo da Internet. Passaram então a fazer parte do cotidiano dos usuários como uma ferramenta que oferece recursos e serviços, permitindo interação e aumento da produtividade, restringindo o compartilhamento dos dispositivos a um aspecto secundário.

Num momento em que as redes se tornam cada vez mais importantes para as empresas, deixando de ser infra-estrutura de comunicação dispensável para se tornarem ferramentas cruciais ao desenvolvimento empresarial, faz-se necessário seu gerenciamento para que atendam adequadamente as necessidades dos seus usuários.

Neste ambiente, diferentes tipos de atividades estão envolvidas para que os serviços sejam providos com qualidade. O modelo de gerência OSI (*Open System Interconnection*) definido pela ISO (*International Standards Organization*) [ISO/IEC 7498:1984] desmembra as tarefas de gerenciamento de rede em cinco pontos: gerência de falhas, gerência de desempenho, gerência de configuração, gerência de contabilidade e gerência de segurança. Assim, podemos ver que a gerência envolve vários fatores interdependentes, todos os quais devendo ser tratados para assegurar o bom funcionamento da rede e dos serviços oferecidos aos seus usuários. Em detrimento às técnicas tradicionais do gerenciamento *ad hoc* [Rose, 1990], esse conjunto de componentes e os problemas associados somente serão gerenciados se uma estrutura bem definida for seguida.

A necessidade de mecanismos de gerenciamento nas redes baseadas em TCP/IP é atendida pelo padrão Internet (*Internet Standard Network Management Framework*), que utiliza o SNMP (*Simple Network Management Protocol*) [Rose, 1990] em associação com o modelo de informação MIB (*Management Information Base*) [Rose & Mccloghrie, 1990a]. Na Internet, o SNMP alcançou grande sucesso por ter sido o primeiro protocolo de gerenciamento não proprietário, público, fácil de ser implementado, possibilitando o gerenciamento efetivo de ambientes heterogêneos. Surgiram no mercado vários tipos de aplicações baseadas no padrão SNMP, quase sempre dependentes de alguma plataforma de gerência. As soluções tradicionais [Rose & Mccloghrie, 1995; Ghetie, 1997], baseadas em estações de gerência de rede centralizadas, executando em ambientes UNIX e usando uma GUI (*Graphical User Interface*) sofrem de muitos problemas [Corcoran, 1996; Smith, 1996; Andrade & Sauvé, 1998]. Um deles é a falta de mobilidade de interface, pois o software de gerência apenas está disponível através da GUI no console da estação de gerência, dificultando o acesso remoto as aplicações.

Hoje, a tecnologia empregada na Web está sendo usada na solução deste problema. Essa tendência não está sendo aplicada apenas à gerência de redes, mas também em *intranets* como forma de acessar bancos de dados, bases de conhecimentos, aplicações de escritórios, etc. O navegador vem se firmando como cliente universal, através do qual usuários acessam os dados corporativos [Rogers, 1996]. No propósito da gerência de redes, estende-se esta idéia como interface padrão do sistema, que pode usar tecnologias empregadas na Web (HTML dinâmico, *Active* X, a*pplets* Java, *Servlets, Java Sever Pages,* etc.) para torná-la mais interativa e sofisticada.

Podem-se agrupar as soluções de gerência de rede baseadas em Web em três classes: soluções de duas camadas através de que um servidor de HTTP embutido nos dispositivos de rede como substituição do *Telnet* [Hyde, 1996]; soluções de três camadas nas quais o GUI das aplicações é substituída por um browser Web mantendo a estação de gerência de rede [Larsen, 1996]; e soluções com quatro ou mais camadas buscando a coexistência do SNMP com outros tipos de protocolos. Recentemente, apareceram duas iniciativas que poderão coexistir com o SNMP como arquitetura de gerência padrão na Internet, utilizando um modelo de dados orientado a objeto e suporte a vários protocolos de gerência. Estes esforços são chamados *Web-Based Enterprise Management* (WBEM) [DMTF, 1999] e *Java Management Extensions* (JMX) [Sun Microsystems, 1999].

1.1 Objetivos da Dissertação

Investigar como algumas das tecnologias empregadas na *Web* podem se aplicar à gerência de redes de computadores baseadas em TCP/IP buscando resolver o problema da falta de mobilidade das aplicações de gerência. Dentro deste objetivo, propor uma arquitetura baseada em Web para utilização destas tecnologias e implementar uma aplicação de gerência que permita a validação da arquitetura proposta.

1.2 Escopo e Relevância

O Grupo de Redes de Computadores (GRC) do Departamento de Sistemas e Computação (DSC) da Universidade Federal da Paraíba (UFPb) vem atualmente investigando a gerência baseada em Web como uma alternativa para as técnicas de gerência tradicionais. Sua meta é implementar estações de gerência sofisticadas, de baixo custo e alta mobilidade, utilizando para isso uma plataforma comum [Sauvé, 1998].

Entretanto, o GRC não está pronto para desenvolver este projeto, pois não conhece suficientemente as possibilidades tecnológicas envolvidas. O primeiro produto deste esforço foi a implementação de uma aplicação, uma solução baseada em Web que pudesse servir de base para novas idéias. Uma investigação tecnológica para o desenvolvimento desta arquitetura de gerência permitirá encontrar as soluções adequadas para o projeto da plataforma. Esta plataforma visa oferecer uma infra-estrutura para facilitar testes práticos de técnicas de gerência resultantes das pesquisas realizadas pelo GRC.

Possuir o domínio da tecnologia empregada nas soluções permite melhores investigações na pesquisa, já que quaisquer alterações que venham a ser necessárias no software de gerência poderão ser implementadas. Isso não seria possível se utilizássemos plataformas prontas do mercado. Além do mais, tais plataformas são caras, extrapolando os recursos disponíveis no DSC. Várias pesquisas que serão desenvolvidas no futuro poderão ser testadas utilizando esta plataforma como base para implementações.

1.3 Estrutura da Dissertação

No Capítulo 2, apresentamos os conceitos necessários para a compreensão dos aspectos envolvidos no desenvolvimento de aplicações de gerência de rede. Levantamos, neste capítulo, limitações das aplicações de gerência tradicionais e como estes problemas estão sendo abordados naturalmente através de soluções direcionadas para a Web.

No Capítulo 3, especificamos uma aplicação baseada em Web que abrange os problemas apresentados no capítulo anterior. Levantamos seus requisitos e identificamos alguns aspectos arquiteturais importantes, propondo uma forma de integração entre as tecnologias Web atuais para o cumprimento dos requisitos levantados.

No Capítulo 4, aprofundamos o projeto da aplicação, mostrando detalhes importantes de implementação, as classes e os pacotes desenvolvidos. Os diagramas das classes desenvolvidas são mostrados no Apêndice C.

No Capítulo 5, demonstramos a aplicabilidade da solução através de um exemplo prático. Os arquivos de configuração deste exemplo são mostrados nos Apêndices A e B.

No Capítulo 6, apresentaremos as conclusões e sugestões para trabalhos futuros.

O glossário da dissertação é apresentado no Apêndice D.

Capítulo 2

A Gerência de Redes de Computadores

Neste capítulo, forneceremos uma visão geral da gerência de redes. Na seção 2.1, apresentamos um conjunto de conceitos básicos sobre o assunto. Na seção 2.2, descrevemos áreas funcionais onde se enquadram as soluções de gerência. Na seção 2.3, apresentamos uma arquitetura básica de gerência, e mostramos na seção seguinte um exemplo de implementação desta arquitetura através do padrão Internet (*Internet Standard Network Management Framework*). Na seção 2.5, abordamos aplicações e plataformas de gerência. Na seção 2.6, levantamos limitações das aplicações de gerência tradicionais, e na seção 2.7, discutimos como estes problemas estão sendo naturalmente abordados através de soluções direcionadas para a Web.

2.1 O Que é Gerência de Redes

Como mostrado na figura 2.1, uma rede de computadores é formada basicamente por três tipos de dispositivos, a saber:

- **Computadores hospedeiros:** são as máquinas usuárias da infra-estrutura de comunicação responsáveis pela execução das aplicações que oferecem serviços aos usuários da rede, funcionando como um ponto de acesso ao sistema.
- **Equipamentos de comunicação:** são as máquinas utilizadas para fornecer serviços de comunicação mas que não executam aplicações finais. Exemplos de tais máquinas são roteadores, comutadores, concentradores, modems, etc.
- **Canais de comunicação:** utilizados para conectar os equipamentos, permitindo assim o encaminhamento de informação entre hospedeiros.

Figura 2.1: Tipos de Dispositivos de uma Rede de Computadores.

Como qualquer outro sistema, as redes de computadores precisam ser gerenciadas para prover um serviço com qualidade. Gerenciar uma rede de computadores consiste em supervisionar e controlar seu funcionamento para que satisfaça as necessidades dos seus usuários [Sloman, 1994]. Uma gerência eficaz pode gerar significativo acréscimo na produtividade da rede pela melhor utilização dos recursos disponíveis, implicando no aumento da qualidade dos serviços oferecidos aos seus usuários e na redução dos custos de operação e manutenção [Meira & Lages, 1998]. Atualmente, pela crescente importância que as redes adquirirem, esta tarefa vem se tornando cada vez mais indispensável.

2.2 Áreas Funcionais da Gerência de Redes

Em um ambiente de rede, diferentes tipos de atividades estão envolvidas para que as redes de computadores possam prover um serviço com qualidade. Segundo o modelo OSI (*Open System Interconnection*) definido pela ISO (*International Standards Organization)* [ISSO/IEC 7498:1984], estas tarefas estão agrupadas em cinco áreas funcionais distintas:

- **Gerência de falhas:** fornece os mecanismos para a detecção, o isolamento e a correção dos problemas surgidos na rede.
- **Gerência de configuração:** fornece meios para manutenção da estrutura física e lógica da rede. Trata da inicialização, alteração e coleta das informações de configuração, incluindo adição e remoção dos dispositivos.

- **Gerência de desempenho:** fornece meios para avaliar o comportamento dos recursos gerenciados e calcular a eficiência dos recursos de comunicação.
- **Gerência de contabilidade:** fornece meios para medir o consumo dos recursos utilizados pelos usuários, possibilitando inclusive o faturamento dos serviços.
- **Gerência de segurança:** fornece meios para a identificação dos usuários, sua autenticação, a autorização dos serviços, o sigilo da informação e a auditoria.

A divisão das atividades de gerência em áreas funcionais ajuda bastante na modularização das soluções. Porém, é importante salientar que a gerência envolve vários fatores interdependentes, todos os quais devendo ser tratados para assegurar o bom funcionamento da rede e dos serviços oferecidos aos seus usuários.

2.3 Arquitetura Básica de Gerência

Genericamente, uma arquitetura de gerência é baseada em quatro componentes:

- **Estação de gerência:** hospedeiro onde as aplicações de gerência são executadas para monitorar e controlar os elementos gerenciados. O software composto por estas aplicações é normalmente conhecido como gerente.
- **Elemento gerenciado:** qualquer dispositivo que possa ser gerenciado. Chama-se agente ao software em execução nestes dispositivos responsável por controlar e responder aos comandos de gerência enviados pela estação de gerência.
- **Informação de gerência:** descreve o estado da rede.
- **Protocolo de gerência:** usado pela estação de gerência e demais elementos gerenciados para efetuar a troca de informação de gerência, definindo operações de monitoração (para obter dados sobre o funcionamento dos elementos) e operações de controle (para alterar parâmetros na configuração dos elementos).

A figura 2.2 ilustra esta arquitetura mostrando a comunicação na rede entre a estação de gerência (gerente) e os elementos gerenciados (agente). Nesta comunicação, a troca de informação de gerência, bem como as operações de controle sobre os dispositivos são realizadas de acordo com um determinado protocolo de gerência.

Figura 2.2: Arquitetura básica de gerência.

Para implementar a arquitetura descrita pela figura 2.2 e obter uma gerência verdadeiramente integrada, é preciso considerar a heterogeneidade das redes e a variedade dos fabricantes envolvidos. Diferentes padrões surgem ao longo do tempo buscando a melhor integração possível entre as diferentes tecnologias, onde a atenção principal está dirigida para a estrutura, o conteúdo e a manipulação da informação de gerência. Como exemplos de padrões de gerência, podemos citar o padrão OSI, definido pela ISO [ISO/IEC 7498:1984]; o padrão DMI (*Desktop Management Interface*) [DMTF, 1999], liderado pela *Microsoft*; e o padrão Internet [Case *et al*., 1990]. Sem querer desconsiderar a importância dos outros padrões existentes, vamos continuar mostrando a arquitetura de gerência mais utilizada atualmente.

2.4 O Padrão de Gerência Internet

A necessidade de mecanismos de gerenciamento nas redes baseadas em TCP/IP é atendida pelo padrão Internet (*Internet Standard Network Management Framework*), que utiliza o SNMP (*Simple Network Management Protocol*) em associação com o esquema de SMI/MIB (*Structure of Management Information / Management Information Base*). Na Internet, este conjunto alcançou grande sucesso por ter sido o primeiro padrão de gerência não proprietário, público, fácil de ser implementado, possibilitando o gerenciamento

efetivo de ambientes heterogêneos. Tendo como objetivo principal reduzir o impacto causado pelas atividades de gerência no desempenho da rede, este padrão concentra as operações mais especializadas em estações de gerência dedicadas, distribuindo um processamento mínimo para os elementos gerenciados.

Os agentes existentes nos elementos gerenciados permitem o acesso à coleções de variáveis escalares armazenadas em uma base de gerência virtual, sendo as operações de gerência restritas à obtenção e controle dos valores destas variáveis.

A SMI [Rose & McCloghrie, 1990b] especifica regras para nomear cada variável de gerência, uma sintaxe para descrever sua estrutura e regras para codificá-la. As variáveis são descritas através de um subconjunto da notação ASN.1 (*Abstract Syntax Notation One*) [ISO/IEC 8824:1987] e codificadas através de um conjunto básico de regras chamado BER (*Basic Enconding Rules*) [ISO/IEC 8824:1987].

Um módulo MIB [Rose & McCloghrie, 1990a] define uma coleção de variáveis relacionadas. Por exemplo, a MIB-II define um módulo obrigatoriamente encontrado nos dispositivos com suporte ao TCP/IP. Os fabricantes dos dispositivos podem definir seus próprios módulos MIB, agrupando a informação associada a uma tecnologia específica.

O protocolo SNMP (*Simple Network Management Protocol*) [Case et al.,1990] é o mecanismo utilizado para permitir a troca das informações de gerência entre agentes e gerentes na rede. Sendo baseado em um simples mecanismo de busca/alteração, possui cinco primitivas básicas de comunicação definidas:

Figura 2.3: Primitivas básicas de comunicação do protocolo SNMP.

- **set**: permite ao gerente alterar o valor de uma variável presente em um agente.
- **get**: permite ao gerente obter o valor de uma variável presente em um agente.
- **get-next**: permite recuperar iterativa e seqüencialmente a informação mantida pelo agente a partir das respostas obtidas por sucessivas requisições.
- **get-response**: contém a resposta a uma requisição SNMP ou o resultado da operação no caso de uma requisição do tipo **set**.
- **trap**: permite que o agente indique ao gerente alguma anormalidade ou erro ocorrido, sem nenhuma requisição prévia do gerente.

Atualmente, existem três versões deste protocolo. A primeira versão, cujas operações básicas foram comentadas acima, ficou conhecida como SNMPv1 [*Case et al.*, 1990] e está sendo ainda bastante utilizada. A segunda versão, conhecida como SNMPv2 [*Case et al.*, 1993], acrescentou funcionalidades para facilitar a recuperação da informação de gerência nos agentes e a comunicação entre gerentes distintos na rede. A última versão, conhecida como SNMPv3 [*Case et al.*, 1998], preocupou-se com aspectos de segurança, buscando garantir a autenticidade e a integridade das operações de gerência.

Resumindo, o padrão de gerência Internet fornece um mecanismo particular para especificar e manipular a informação de gerência sobre a rede, baseada em uma arquitetura bem tradicional, utilizando estações de gerência de rede centralizadas.

2.5 Aplicações e Plataformas de Gerência

Embora o padrão de gerência Internet forneça um mecanismo de instrumentação para gerência, falta ainda acrescentar a *inteligência* necessária para se realizar as atividades de gerência. Esta capacidade é embutida na estação de gerência através das aplicações de gerência, que utilizam os dados coletados para realizar comparações, manipulações e análises. Os resultados são mostrados na interface para o usuário como gráficos, alarmes, sinais sonoros ou outros recursos que indiquem o estado da rede, erros ou anormalidades.

Uma estação de gerência pode executar várias aplicações de gerência, onde uma poderia se especializar em gerência de faltas, outra na gerência de desempenho para planejamento de capacidade, uma terceira ainda na configuração de roteadores, etc. Porém, devemos lembrar que essas várias aplicações terão coisas em comum.

Figura 2.4: Aplicações de Gerência de Rede.

Por exemplo, todas devem saber a topologia da rede, possuir mecanismos de conversa com agentes através do protocolo de gerência, ter formas simples de apresentar informação gráfica ao usuário, etc. Dessa forma, uma **plataforma de gerência** é um software especial instalado na estação de gerência que agrega toda a funcionalidade comum a várias aplicações de gerência, incluindo descobrimento automático da topologia da rede, rotinas de acesso às MIBs dos agentes utilizando o protocolo de gerência, armazenamento das informações obtidas em um banco de dados, etc.

Além destas funções, uma plataforma deve prover uma GUI (*Graphical User Interface*) que permita o usuário atuar sobre a rede, oferecendo integração visual entre as várias aplicações de gerência. Um outro componente importante é a API (*Application Programming Interface*) oferecida para possibilitar o desenvolvimento aplicações mais rapidamente. *Open View* da *Hewlett Packard*, *NetView* da IBM e *Spectrum* da *Cabletron Corp* são alguns exemplos de plataformas comerciais disponíveis no mercado.

2.6 Limitações da Arquitetura de Gerência Tradicional

As soluções de gerência tradicionais baseadas no padrão Internet alcançaram um grande sucesso na década passada. Porém, esta arquitetura não é mais suficiente para oferecer soluções com o nível de qualidade atualmente necessário, impondo sérias limitações ao desenvolvimento das atividades de gerência [Huntington-Lee, 1999]. Vamos então analisar e propor soluções para alguns dos problemas básicos desta arquitetura, agrupados em quatro áreas: integração, mobilidade, segurança e custos.

2.6.1 Integração

Uma das principais tarefas de uma plataforma de gerência seria a de prover uma maneira para integrar e apresentar as aplicações mediante uma única GUI. A estação de gerência geralmente executa várias aplicações diferentes, e essas aplicações deveriam alcançar um alto nível de integração. Porém, esta integração não é tão simples e muitos fabricantes optam por não suportá-la em seus produtos [Andrade e Sauvé, 1998].

Em particular, as questões relativas à interface das aplicações poderiam ser bem resolvidas com o uso da tecnologia Web. Atualmente, o browser já é considerado o cliente universal, uma interface primária a partir da qual o usuário acessa dados [Rogers, 1996]. No propósito da gerência de redes, estender esta idéia para a interface padrão do sistema seria bastante natural. Várias tecnologias empregadas na Web como HTML dinâmico, ActiveX, Applets Java, Servlets e JSP (*Java Server Pages*) poderiam ser usadas para construir interfaces interativas e sofisticadas.

2.6.2 Mobilidade

Na arquitetura tradicional, as aplicações estão disponíveis apenas a partir do console da estação de gerência, ou no máximo a partir um terminal X-Window situado na mesma rede local. Esta falta de mobilidade de interface fere seriamente um dos requisitos para as tarefas de gerência: gerentes de rede desejam acessar suas aplicações de gerência a partir de qualquer ponto da rede ou mesmo de qualquer ponto fora da rede (via Internet ou Intranet) e não apenas a partir de um console da estação de gerência.

Usando um browser como interface, pode-se resolver facilmente este problema, gerenciando a rede a partir de qualquer máquina da rede ou ligada à Internet. Além disso, com o browser ganha-se ainda a independência de sistema operacional na máquina cliente, onde é executada a interface GUI.

2.6.3 Segurança

Soluções de gerência baseadas no padrão Internet provêem muito pouca segurança. A segurança é obtida basicamente através de uma senha, chamada de *Community Name*, enviada sem nenhum recurso de criptografia em cada PDU (*Protocol Data Unit*) do SNMP. Tentativas foram realizadas para melhorar este aspecto pelas versões posteriores, porém não foram bem sucedidas e na prática não estão sendo utilizadas [Rogers, 1996].

Esta falta tem restringido a utilização da solução SNMP para o monitoramento das redes e não para o seu controle. O controle das redes (alterações na configuração dos

dispositivos) é normalmente feito através de uma sessão *Telnet* com o dispositivo gerenciado, ou mais recentemente através de aplicações com servidor Web embutidos. Esta última vem sendo bastante utilizada e está quebrando o *deadlock* existente entre fabricantes de dispositivos e desenvolvedores [Andrade e Sauvé, 1998].

A solução para este problema também pode ser encontrada na Web. Os serviços que exigem um nível maior de segurança, principalmente o comércio eletrônico, motivou o aparecimento de boas soluções neste setor como o S-HTTP (*Secure* HTTP). Assim, ao invés de procurar novas soluções, parece ser mais eficiente utilizar um protocolo que já tenha implementado eficazes recursos de segurança, como é o caso do HTTP.

2.6.4 Custos

Uma razão para os elevados custos associados à gerência é o elevado preço das plataformas e aplicações de gerência: tais produtos em conjunto facilmente alcançam dezenas de milhares dólares. A estes custos, ainda são somados gastos envolvidos no controle de versões e distribuição dos softwares e treinamento dos usuários [Rose, 1996].

Todos estes custos associados à compra e manutenção de soluções para gerência e no treinamento para seu uso, resultam no fato de que pequenas empresas ficam de fora deste mercado e não investem em soluções de gerência, confiando apenas em técnicas *ad hoc*.

Utilizar um browser Web como interface do sistema seria um fator de redução de custos, visto que este software é praticamente gratuito e de uso bastante intuitivo, o que alivia os gastos com o treinamento de usuários. Ainda mais, isto facilitaria o acesso às aplicações embutidas nos dispositivos de rede, possibilitando uma redução nos custos de tais aplicações. Outra possibilidade seria o *download* automático de código nos dispositivos providos com JVM (*Java Virtual Machine*), mais um golpe nos custos envolvidos com a gerência. No entanto, não está claro como o alto custo das plataformas poderia ser afetado por estas mudanças; contudo, a entrada de novos competidores no cenário da gerência de redes poderá contribuir para a redução dos preços destes *softwares*.

2.7 Arquiteturas Baseadas em Web

Observando as soluções propostas para os problemas das aplicações de gerência tradicionais, percebemos que as tecnologias Web desempenham um papel importante. Suas contribuições vão desde a possibilidade de efetuar comunicação segura via S-HTTP, até a presença de servidores Web, *browsers*, *applets* e *servlets* nestas soluções. Esta tendência se

tornou tão forte atualmente, que estas soluções ficaram conhecidas como arquiteturas de gerência baseada em Web. Passamos a discutir tais arquiteturas nas próximas seções.

2.7.1 Arquitetura em Duas Camadas

A Arquitetura em Duas Camadas é a mais simples das alternativas para a gerência baseada em Web. Nesta arquitetura um servidor HTTP é embutido nos dispositivos de rede (roteadores, comutadores, etc.) possibilitando que através de páginas HTML, este dispositivo seja monitorado ou controlado (Figura 2.5).

Figura 2.5 - Arquitetura Em Duas Camadas

Cada dispositivo tem sua própria URL principal e pode ser gerenciado através de um browser Web. As páginas HTML de um determinado dispositivo podem incluir formulários processados por programas CGI ou outra técnica que possibilite a configuração destes dispositivos via HTTP. A principal característica desta arquitetura é que ela é uma substituição ou pelo menos uma nova alternativa ao *Telnet* nas atividades de configuração. Ao invés de abrir uma sessão *Telnet* com o dispositivo e utilizar uma interface a caractere para realizar as alterações nos parâmetros de configuração, o usuário acessa via browser as páginas HTML do dispositivo e, através do preenchimento de formulários, realiza a sua configuração.

Não há dúvidas de que utilizar páginas HTML para configuração de dispositivos é muito mais amigável e simples do que utilizar uma interface a caractere como o *Telnet*. A relação custo-benefício desta abordagem é tão positiva que fabricantes de dispositivos já estão embutindo servidores HTTP em seus produtos, visando obter um diferencial em relação à concorrência. A queda nos preços do hardware e a escala de produção dos dispositivos faz com que o acréscimo do servidor HTTP não represente um fator proibitivo para a comercialização destes dispositivos.

Observe que uma arquitetura em duas camadas não impede o uso de outras soluções de gerenciamento tais como o SNMP. O dispositivo é simplesmente equipado com suporte ao SNMP e ao HTTP. Portanto soluções baseadas no SNMP podem ser usadas sem nenhuma mudança. Então o que nós ganhamos? Uma melhor interface, mobilidade, segurança e baixos custos. Também houve uma mudança nos incentivos, desde que qualquer fabricante de hardware para rede poderá embutir nele aplicações que não necessitem de suporte a diferentes plataformas.

Por outro lado, esta arquitetura *não* representa uma solução definitiva para a resolução dos problemas de gerência. Embora seja muito fácil navegar pelos dispositivos de rede e configurá-los através de páginas HTML, sabemos que gerenciar uma rede envolve muito mais que estas tarefas. Como poderíamos integrar informações escondidas dentro de centenas de páginas HTML para obter uma visão geral da rede?

Nós precisamos de correlações entre eventos que tenham ocorrido em diferentes dispositivos. Precisamos planejar e executar ações de gerência baseada num conjunto de dispositivos e condições. E mais importante ainda, precisamos automatizar a gerência.

Resumindo, embora seja claro que a arquitetura em duas camadas oferece uma boa substituição à solução oferecida anteriormente pelo *Telnet* e representa uma forte tendência entre os fabricantes de dispositivos, parece que apenas empresas com redes muito pequenas se contentariam em utilizar apenas esta arquitetura para gerenciar suas redes.

2.7.2 Arquitetura em Três Camadas

Como na arquitetura em duas camadas, a arquitetura em três camadas também utiliza o SNMP e o HTTP simultaneamente. Porém, estes dois protocolos são usados de forma diferente, como mostrado na figura 2.6. A estação de gerência comunica-se com os agentes utilizando o SNMP da mesma maneira como é feito na abordagem tradicional de gerência.

Nesta estação de gerência existe um servidor HTTP embutido, permitindo que um browser possa ser utilizado para acessar a plataforma ou aplicação presente na estação de gerência. Observe que, enquanto a solução em duas camadas é implementada pelos fabricantes de dispositivos, a arquitetura em três camadas representa uma evolução alcançada pelos desenvolvedores de plataformas e aplicações de gerência.

Figura 2.6 - Arquitetura em Três Camadas

Esta arquitetura provê uma mesma interface para todas as aplicações, resolve o problema da falta de mobilidade, reduz os custos e permite que múltiplos browsers acessem as informações de gerência simultaneamente. Outra vantagem desta arquitetura é que ela é capaz de apresentar a visão geral da rede ao contrário da arquitetura em duas camadas apresentada anteriormente. Parece claro que as arquiteturas em duas e três camadas podem ser (e normalmente são) usadas simultaneamente.

2.7.3 Outras Iniciativas

Examinando o futuro na área da gerência baseada em Web, consideramos duas iniciativas recentes que buscam coexistir com o SNMP como arquitetura de gerência padrão na Internet, utilizando um modelo de dados orientado a objeto e suporte a vários

protocolos de gerência. Estes esforços são chamados *Web-Based Enterprise Management* (WBEM) [DMTF, 1999] e *Java Management Extensions* (JMX) [Sun Microsystems, 1999]. O WBEM é um padrão desenvolvido pela *Distributed Management Task Force* (DMTF) [DMTF, 1999], uma organização da indústria liderada pela *Microsoft* que está conduzindo o desenvolvimento, adoção e unificação de padrões de gerência de rede. JMX consiste em um jogo de especificações e ferramentas desenvolvidas pela Sun Microsystem para construção de soluções de gerência utilizando a tecnologia Java.

Assumindo um browser como software cliente básico, embora estas iniciativas não restrinjam o uso de outras alternativas, estas duas arquiteturas vão adiante e tentam resolver as questões com respeito à escalabilidade, modelo de dados extensível, integração entre as aplicações e segurança. Uma vez que estas novas abordagens tentam resolver os desafios da gerência, elas propõem mudanças estruturais no modelo de dados e no protocolo utilizado. Por isso é importante sua padronização antes de serem realmente empregadas em larga escala.

Capítulo 3

Uma Aplicação de Gerência de Rede Baseada em Web: Requisitos e Projeto Arquitetural

Como vimos no capítulo anterior, as aplicações de gerência tradicionais sofrem de certos problemas que dificultam o trabalho dos gerentes e administradores de redes de computadores. Visando oferecer uma solução adequada às necessidades desses usuários, descrevemos neste capítulo a especificação de uma aplicação baseada em Web que abrange os problemas apresentados na seção 2.6.

O desenvolvimento desta aplicação se baseia nas etapas básicas de qualquer processo de desenvolvimento de software [Rumbaugh *et al.*, 1999]: análise do domínio do problema, projeto da solução, implementação e testes. Na seção 3.1, iniciamos o entendimento do domínio da problema examinando um protótipo da aplicação. Na seção 3.2, levantamos os requisitos que devem ser atendidos pela solução. Na seção 3.3, apresentamos o projeto arquitetural da solução, propondo uma forma de integração entre as tecnologias Web atuais para o cumprimento dos requisitos levantados. Detalhes de implementação e testes serão abordados no capítulo 4.

3.1 Protótipo da Aplicação

Uma vez que seja necessário propor uma solução para qualquer problema, o primeiro passo é entender bem o problema. Para isso, é necessário identificar os requisitos que devem ser atendidos pela solução. Uma importante técnica para levantamento e validação de requisitos é a utilização de protótipos [Sommerville, 1994]. Este tipo de abordagem é

uma forma de conter a dificuldade que os usuários possuem para formular e entender especificações estáticas, bem como identificar os riscos envolvidos no projeto.

Nesta seção, apresentaremos o protótipo desenvolvido pelo professor Jacques Phillipe Sauvé (DSC/UFPB) [Sauvé, 1999]. Este protótipo foi utilizado para o levantamento dos requisitos da solução junto aos administradores e gerentes das redes de computadores. Todo o conteúdo desta seção se baseia na fonte acima referenciada, onde o objetivo é aproveitar a experiência adquirida, visando obter um ponto de partida para a experimentação de novas idéias e tecnologias.

3.1.1 Requisitos

Descrevemos a seguir os objetivos propostos para o protótipo da solução, projetada como uma aplicação básica de gerência para atender os seguintes requisitos iniciais:

- A solução deve ser uma aplicação de gerência e não uma plataforma;
- A solução deve usar a arquitetura de gerência padrão da Internet, dando suporte ao SNMP, versões 1 e 2;
- A interface da aplicação deve ser acessada apenas através de um browser Web*;*
- A interface permite a navegação através da rede usando mapas hierárquicos sensíveis à posição do clique do mouse;
- Os mapas da rede precisam mostrar o estado de cada parte da rede através de um mecanismo visual apropriado, como um código de cor, permitindo uma rápida visão e localização da presença de qualquer problema na rede;
- Suporte para MIB, acessando as estatísticas históricas através de gráficos diários. Apenas a monitoração (e não a alteração) das variáveis MIB deve ser permitida, devido à fraca segurança da arquitetura SNMP;
- Os gráficos apresentados na interface podem ficar desatualizados dentro de um determinado período de tempo, mas existirá um mecanismo para obter estatísticas atualizadas com um tempo de resposta de no máximo cinco minutos para redes com até dezenas de equipamentos gerenciados;
- Devem ser providos alarmes configuráveis, baseados em um mecanismo capaz de descobrir valores críticos para as variáveis MIB;

- A configuração pode ser manual, através de um editor de texto. A solução deve ser dirigida a dados (*data-driven*), sem informação de topologia ou elementos visuais codificados no programa. Isto inclui páginas HTML e gráficos, etc.

3.1.2 Especificação Funcional

Após sua instalação e configuração, o protótipo da aplicação pode mostrar a rede graficamente através de uma interface Web. A situação da rede é indicada utilizando cores: um ponto verde sobre um equipamento ou canal de comunicação significa um estado operacional correto; pontos pretos, amarelos e vermelhos indicam problemas de diferentes níveis críticos, que podem ser propagados através de mapas hierárquicos (figura 3.1).

Figura 3.1: Visão geral da rede gerada pelo protótipo

Clicando sobre as figuras com indicações coloridas, o operador da estação de gerência pode se deparar com sub-mapas ou acessar informações sobre o comportamento dos equipamentos durante a semana, ou em outra data específica (figura 3.2). Preenchendo um pequeno formulário existente nas páginas HTML dos equipamentos, o operador pode

remeter pedidos ao servidor Web, que invoca um programa CGI (*Common Gateway Interface*) para atualizar os gráficos e estados dos equipamentos gerenciados.

Figura 3.2: Informações obtidas sobre um canal de comunicação.

Os valores representados nos gráficos foram obtidos utilizando o protocolo de gerência SNMP. Essas informações, expostas graficamente, são úteis para detectar o tipo de problema que está acontecendo na rede ou até mesmo prever quando esses problemas vão acontecer. Por exemplo, observando a taxa de entrada e saída de um canal durante o decorrer do tempo (meses, ou até mesmo semanas), será possível prever quando esta taxa ultrapassará um valor crítico e, desta forma, providenciar a contratação de um canal mais veloz, antes de enfrentar uma degradação sensível de desempenho.

3.1.3 Arquitetura

A partir da especificação e dos requisitos levantados, a arquitetura do protótipo foi estruturada em três módulos principais, escritos em linguagem *Perl*, utilizando também outros programas auxiliares para realizar seu trabalho (figura 3.3). Um único arquivo de configuração precisa ser editado, definindo as configuração necessárias para os três módulos da aplicação. Neste arquivo estão reunidas informações sobre os equipamentos

gerenciados (endereço de rede, intervalo de monitoração, variáveis SNMP monitoradas, etc.) e sobre as páginas HTML da rede (hierarquia de mapas, figuras, gráficos, etc.).

O módulo denominado Monitor é encarregado de coletar periodicamente informações de gerência junto aos agentes SNMP e guardá-las apropriadamente em um arquivo de dados. Ele também verifica se as variáveis SNMP ultrapassaram um determinado valor crítico, podendo enviar alarmes via *e-mail* (ou outro mecanismo) quando isto acontece. Outro módulo, denominado Construtor, apanha informações do arquivo de dados para gerar as GIFs (figuras, gráficos e mapas da rede), arquivos MAP (para imagens sensíveis à posição do mouse) e construir as páginas Web a partir dos modelos HTML contendo código *Perl* embutido. Uma razão forte para o uso de modelos é que páginas de equipamentos parecidos (considere dois roteadores iguais) podem ser produzidas a partir de um único modelo.

Figura 3.3: Arquitetura do Protótipo da Aplicação

O uso de modelos foi impulsionado pela exigência por uma solução orientada a dados, onde esses dados são obtidos de três lugares: do arquivo de configuração, dos modelos HTML e dos arquivos de dados. O módulo Construtor processa todas estas

entradas para produzir as páginas HTML finais. Como exemplo do uso de modelos HTML, o seguinte arquivo é utilizado para produzir todos os mapas da rede (figura 3.4):

```
<HTML>
<HEAD><TITLE>WEBMANAGER - ZZZ$ObjectName{$Object}ZZZ </TITLE></HEAD>
<BODY>
<B><FONT SIZE=5>
<P ALIGN="CENTER"> WEBMANAGER - ZZZ$ObjectName{$Object}ZZZ </P>
</FONT></B>
<FORM ACTION= ”/cgi-bin/webmngr.pl.cgi” METHOD=”Post” NAME=”Update”>
<STRONG>Atualizada em </STRONG>:ZZZ&FormatDateAndTime($DateNow)ZZZ
<INPUT TYPE=“submit” NAME=“UpdateNetMap” VALUE="Update">
<INPUT TYPE=“hidden” NAME=“Object” VALUE=”ZZZ$ObjectZZZ”>
</FORM>
<P ALIGN = “CENTER”> ZZZ&BuildMap($Object)ZZZ
<A HREF= “ZZZ$ObjectZZZ.map”>
<IMG SRC= “ZZZ$ObjectZZZ.gif” ismap>
</A></P>
</BODY>
</HTML>
```

Figura 3.4: Modelo HTML dos mapas da rede

Um módulo CGI também faz parte da estação de gerência, adicionando interatividade. É através dele que os estados dos equipamentos, gráficos, etc. podem ser atualizados manualmente através de pedidos realizados a partir da interface do usuário. O CGI trabalha simplesmente executando os outros dois módulos da aplicação, sendo esta a única razão para que todos sejam executados na mesma máquina. Caso contrário, como a comunicação entre os módulos é realizada através de arquivos, estes programas poderiam rodar em máquinas separadas usando um sistema de arquivos distribuído.

3.1.4 Análise do Protótipo

O protótipo da aplicação atendeu a todos os requisitos propostos anteriormente, sendo utilizada para gerenciar várias redes (as redes dos sete campi da Universidade Federal da Paraíba, o POP-Pb, parte da Rede Nacional de Pesquisa, etc.). Algumas dessas

redes envolvem dezenas de equipamentos gerenciados. Como pontos mais bem sucedidos dentro da sua arquitetura podemos destacar:

- **Linguagem** – como linguagem usada para implementação da ferramenta, *Perl* é excelente para lidar com aplicações envolvendo texto, além de facilitar o processo de codificação (seu desenvolvimento levou pouco mais de um mês);
- **Alarmes** - muitos dos problemas existentes na rede podem ser identificados sem intervenção do usuário. Gastando algum tempo na configuração, uma administração razoavelmente automática pode ser alcançada, especialmente com respeito à gerência de faltas e desempenho;
- **Configuração** - modelos HTML expandidos usando código *Perl* embutido facilitou bastante o processo de configuração da aplicação, permitindo o uso de páginas padronizadas para cada modelo de equipamento gerenciado;
- **Interface** - com respeito à velocidade, usar um browser Web para acessar páginas HTML (mapas e imagens) permite uma visualização rápida, mas pode sobrecarregar a estação de gerência durante a geração dessas informações. Outra vantagem é que esta interface permite acesso remoto de qualquer ponto da rede;

Destacamos agora os principais problemas identificados no protótipo. Esses problemas restringem o uso do protótipo para redes de pequeno porte, com até algumas dezenas de elementos gerenciados. Certamente, estes serão os principais pontos abordados para a produção de novas versões da aplicação:

- **Segurança** - nenhum aspecto de segurança foi considerado dentro dos requisitos;
- **Estrutura Hierárquica** - um problema foi a imposição de uma estrutura hierárquica para os objetos administrados. Um grafo geral teria sido mais apropriado, permitindo a presença de um objeto em diferentes mapas da rede;
- **Compilador MIB** - a falta de um compilador MIB força a definição de muita informação sobre as variáveis MIB no arquivo de configuração. Este compilador também poderia ser usado na implementação de um MIB *browser*, um pequena aplicação que permite visualizar as variáveis dos equipamentos;

- **Gráficos Estatísticos** - os gráficos estatísticos geram curvas diretamente dos valores obtidos das variáveis MIB. Um mecanismo para permitir cálculos gerais envolvendo essas variáveis deveria ter sido implementado;
- **Modelos HTML** - a utilização de código *Perl* nos modelos HTML deve ser trocada por mecanismos padronizados ao invés da solução desenvolvida;
- **Configuração** - a configuração da aplicação para uma determinada topologia particular de rede é muito demorada. Este problema deve-se tanto à edição manual de um arquivo texto quando à fraca verificação de erros pela ferramenta;
- **Alarmes –** a aplicação não realiza a correlação de alarmes. Como exemplo, se um roteador sofrer uma pane, vários alarmes iriam ser disparados ao mesmo tempo (roteador e interfaces), sem a indicação de uma causa comum;
- ***Traps*** - nenhum suporte a *traps* SNMP foi provido, e isto poderia ser usado para minimizar o tráfego gerado pelo protocolo SNMP na rede;
- **Escalabilidade** – à medida que novos equipamentos monitorados são adicionados à rede gerenciada, o tempo de processamento para atualização do estado dos mapas pode alcançar um valor insuportável para o usuário (vários minutos);
- **Desempenho** - uma avaliação da velocidade da ferramenta indicou que a atualização do estado dos mapas da rede através do CGI pode levar dezenas de minutos para redes com mais de centenas de equipamentos gerenciados;
- **Atualidade da Informação** – a informação mostrada na interface não reflete bem o estado atual da rede. Mesmo quando os mapas acabaram de ser gerados, algum problema na rede pode ter acontecido durante a construção das páginas;
- **Interatividade** – a interface provê pouca interatividade, principalmente com respeito à escolha de variáveis, escala de tempo e estilo dos gráficos. Na solução atual, apenas é utilizado CGI para interagir com o usuário;

3.2 Levantamento dos Requisitos

O objetivo agora é descrever os requisitos da solução, de forma a melhorar suas características e funcionalidades, mas conservando alguns pontos fortes encontrados na

versão anterior. A partir do estudo do protótipo apresentado, e com base nos problemas expostos na seção 2.6, seguem os requisitos a serem atendidos em próximas versões.

3.2.1 Requisitos Funcionais

A lista de requisitos a seguir descreve as funcionalidades que devem estar disponíveis para o usuário nas próximas versões da aplicação:

Gerência de Desempenho

F1. Monitoração da Rede - levantamento periódico do estado dos equipamentos gerenciados, com intervalo de tempo configuráveis para obtenção da informação de gerenciamento nos agentes espalhados na rede;

F2. Notificações - deve ser oferecido suporte à notificação automática de eventos[1] pelos agentes espalhados na rede. Agentes podem monitorar as variáveis MIB que estão sob seu controle, de acordo com parâmetros previamente configurados no equipamento gerenciado. Em função do comportamento das variáveis, o agente pode enviar um *trap* (comando SNMP), notificando um evento ao gerente;

F3. Planejamento de Capacidade - deve permitir acesso às estatísticas históricas, provendo a base de dados necessária para possibilitar um conhecimento mais preciso sobre a utilização dos recursos disponíveis. O período máximo de armazenagem dos dados deve ser configurado pelo usuário;

F4. Recuperação da Informação - a recuperação da informação de gerenciamento deve ser realizada diretamente dos valores obtidos das variáveis MIB ou através de cálculos gerais envolvendo as variáveis;

Gerência de Falhas

F5. Alarmes - deve ser provido um mecanismo utilizando níveis de criticalidade para informar ao usuário ocorrências sobre o comportamento da rede, de acordo com os valores obtidos das variáveis MIB implementadas nos equipamentos gerenciados;

[1] Um evento é um momento interessante de atividade na rede (por exemplo, o cruzamento de um limiar, indicando que a taxa de erros ultrapassou um determinado valor).

F6. *Log* de Alarmes - os alarmes já ocorridos na rede devem ser armazenados para consulta posterior de acordo com seu tempo de ocorrência;

F7. Correlação de Alarmes - a solução deve realizar correlação para identificar padrões ou tendências nos alarmes gerados, de forma que múltiplas manifestações relacionadas possam ser tratadas como uma unidade. Por exemplo, se um roteador sofre pane, apenas um alarme geral deve ser gerado, para reduzir a quantidade de ocorrências apresentadas e disponibilizar informações realmente relevantes para o usuário;

F8. Ações – deve existir uma forma de executar e configurar ações programadas pelo usuário quando um determinado alarme for detectado;

F9. Relatórios - deve haver facilidades para geração de relatórios, incluindo a geração e o envio automático via e-mail;

Gerência de Configuração

F10. Parametrização dos Equipamentos - deve dar suporte à configuração remota dos equipamentos, utilizando programas ou métodos mais seguros que a solução provida pelo SNMP (primitiva *set*);

F11. Autotopologia - deve ser provido um mecanismo para reconhecimento automático da configuração da rede, reduzindo a necessidade de configuração manual. Três aspectos devem ser considerados: descobrimento de equipamentos presentes na rede, identificação dos canais de comunicação entre estes equipamentos e geração automática dos mapas para visualização da rede;

Interface

F12. Representação da Rede - de modo confortável, a aplicação deve mostrar graficamente a rede gerenciada na interface do usuário, sem sobrecarregar a tela com objetos visuais;

F13. Estado dos Equipamentos - a solução deve indicar a situação dos equipamentos gerenciados através de um mecanismo visual que permita a utilização de níveis de criticalidade;

F14. Informação sobre Equipamentos - deve existir uma forma de acessar atributos sobre um equipamento gerenciado (descrição, endereço de rede, etc.);

F15. Indicação de Problemas na Rede - deve existir um ponto da interface para relatar todos os problemas que estão acontecendo na rede, podendo-se acessar diretamente as informações sobre os equipamentos envolvidos;

F16. Resumo Operacional – num mesmo ponto da interface deve ser possível visualizar estatísticas selecionadas de diferentes equipamentos;

F17. Gráficos Estatísticos - a visualização da informação de gerência coletada deve ser apresentada através de diferentes tipos de gráficos estatísticos, facilitando a avaliação do comportamento dos recursos;

F18. MIB *Browser* - através de uma aplicação simples, deve ser possível visualizar qualquer variável MIB de um determinado equipamento;

Acesso à Aplicação

F19. Permissões de Acesso - devem existir mecanismos para controlar o acesso dos usuários à aplicação. Neste sentido, as seguintes permissões de acesso poderão ser atribuídas aos usuários:

(a) Usuário do Sistema - permite acesso para visualização das páginas da rede e para as aplicações de monitoração dos equipamentos;

(b) Administrador do Sistema - permite uma manipulação completa da aplicação, incluindo configuração e o controle dos usuários.

F20. Atribuição de permissões - permissões devem ser associadas aos usuários. A solução deve possui um conjunto inicial de usuários cadastrados cujos nomes, *login* e permissões de acesso devem ser informados;

F21. Identificação dos Usuários - para acessar a aplicação, o usuário deverá fazer um *login*, identificando-se e fornecendo senha correta. Receberá então as permissões apropriadas. Para acesso a configuração da aplicação deve ser verificada novamente a permissão do usuário.

Configuração

F22. Configuração dos Usuários - deve-se poder adicionar ou excluir usuários, bem como editar seus atributos (identificação, senha e permissão);

F23. Configuração das MIBs – deve-se poder inserir novos módulos MIBs (SMI versão 1 e 2) ao conjunto de módulos distribuído com a aplicação, bem como excluir os módulos já existentes;

F24. Configuração da Rede - através de um mecanismo automático deve se possível reconhecer os equipamentos existentes na rede, bem como sua topologia. A partir do reconhecimento inicial, deve ser possível organizar uma estrutura lógica para apresentação gráfica da rede através da interface *Web*;

F25. Configuração dos Equipamentos - deve ser permitido excluir ou inserir novos elementos não identificados pela autotopologia, bem como editar seus atributos (nome, endereço de rede, responsável, etc.). Os elementos devem ser empacotados com toda a informação de configuração de modelos (incluindo MIBs, figuras, etc.);

F26. Configuração de Alarmes - deve se permitido relacionar alarmes às variáveis dos equipamentos, podendo-se também configurar seu nível de criticalidade e ações associadas;

F27. Configuração dos Gráficos – deve ser possível configurar os seguintes atributos: intervalo de amostragem, tipo (barra ou temporal), dimensão, cores, tamanho, escala vertical (domínio da variável), escala horizontal (hora) e data.

F28. Configuração dos Resumos Operacionais - deve ser possível configurar quais os equipamentos envolvidos no resumo operacional e quais as variáveis ou estatísticas visualizadas para os equipamentos selecionados;

3.2.2 Requisitos Não-Funcionais

A lista de requisitos não-funcionais a seguir também são impostas à solução.

Mecanismos de Instrumentação

A solução de gerência deve dar suporte à arquitetura de gerência padrão da Internet, onde as seguintes versões destes padrões devem ser suportadas:

NF1. Protocolo de Gerência – deve prover suporte ao protocolo SNMP, versões 1, 2 e 3;

NF2. Informação de Gerência – deve prover suporte para qualquer MIB compatível com a SMI, versões 1 e 2;

Segurança

Como mecanismo de segurança, o conceito de comunidade é empregado pelo protocolo SNMP para determinar a relação entre agentes e um ou mais gerentes, permitindo implementar políticas de acesso para as variáveis monitoradas. A aplicação deve possuir o seguinte requisito quanto ao uso de comunidades:

NF3. Comunidade SNMP - a aplicação deve suportar a definição de várias comunidades SNMP para os equipamentos gerenciados;

Arquitetura

NF4. Arquitetura *Web* - aplicação deve utilizar uma arquitetura baseada em tecnologia *Web* para gerenciar redes de computadores baseadas em TCP/IP;

NF5. Aplicação Centralizada – a solução continua sendo uma aplicação centralizada, não hierárquica, sem distribuição das tarefas entre estações espalhadas na rede. Ainda não se pretende criar uma plataforma de gerência;

Plataformas operacionais

NF6. Primeira Camada – deve ser possível gerenciar qualquer tipo de dispositivo com suporte ao SNMP em qualquer versão do protocolo;

NF7. Segunda Camada - a estação de gerência deve possuir sistema operacional UNIX (qualquer versão) ou Windows 98/NT, podendo utilizar qualquer servidor Web;

NF8. Terceira Camada - é permitido exigir recursos avançados que se apliquem aos dois browsers Web mais populares do mercado (como suporte ao JDK 1.2), podendo existir um *firewall* entre o cliente e a estação de gerência;

Desempenho

NF9. Atualidade da Informação – a informação mostrada na interface deve refletir o estado da rede, com tempo de latência de segundos para redes com até algumas dezenas de equipamentos gerenciados, até minutos para redes maiores;

NF10. Indicação de Falhas – o tempo de latência entre uma falha na rede e sua indicação visual pode variar de valores entre segundos para redes com dezenas de equipamentos gerenciados, até minutos para redes maiores;

NF11. Visualização dos Gráficos – A visualização de um gráfico não deve demorar mais do que 2 segundos numa rede local Ethernet de 10 Mbps. Para gráficos com mais de 4 estatísticas, este tempo não pode ser maior que 5 segundos;

Documentação

NF12. Manual do Usuário - um manual deve ser gerado durante o processo de desenvolvimento, devendo ser distribuído em HTML junto com a aplicação;

NF13. Guia de Instalação - além do manual de utilização, deve também ser gerado, ao final do desenvolvimento, um completo guia de instalação;

Instalação

NF14. Modo de Instalação - os componentes da ferramenta devem ser empacotados num único módulo, com sua instalação podendo ser realizada em modo visual ou em linha de comando. Nenhum requisito de desinstalação é imposto para aplicação.

Os requisitos acima especificados, junto com os requisitos funcionais apresentados na seção anterior, são os pontos básicos que devem ser atendidos de modo a satisfazer as necessidades dos administradores e gerentes de redes de computadores. Certamente, uma solução final exige uma lista de requisitos mais complexa que considere, por exemplo, outros aspectos relacionados a desempenho ou segurança. Neste trabalho, entretanto, concentramo-nos nos requisitos listados na certeza de que o seu atendimento já caracteriza bem a solução a ser fornecida. A partir da próxima seção, é apresentada uma solução que atende a maioria de tais requisitos (mas não todos).

3.3 O Projeto Arquitetural

Durante esta fase, buscaremos listar os maiores problemas a serem solucionados, e as possibilidades tecnológicas para cada problema. Isto é o que chamamos de processo *risk confronting*, um processo que auxilia no estabelecimento da arquitetura base da solução.

Devido à natureza da aplicação, alguns aspectos devem ser considerados na seleção das tecnologias utilizadas. Buscando facilitar o trabalho de desenvolvimento, bem como garantir o cumprimento dos requisitos propostos para a solução, as tecnologias escolhidas devem atender principalmente às seguintes características:

- **Portabilidade** - Um ambiente de rede envolve diferentes tipos de equipamentos, com diferentes sistemas operacionais. Assim, as ferramentas deve ser bastantes flexíveis, facilitando a portabilidade da aplicação para diferentes plataformas;
- **Voltada para redes -** Amplamente distribuída, a aplicação possui componentes espalhados em diferentes máquinas. Para permitir a interação entre estes componentes, as tecnologias devem oferecer amplo suporte a rede;
- **Integração com a Web** - As tecnologias devem ser compatíveis com aquelas empregadas na Web, pois alguns módulos da aplicação serão invocados através do browser para execução no servidor ou na máquina cliente.

Existem poucas soluções disponíveis, respeitando simultaneamente essas exigentes características. Basicamente existem duas opções: aceitar as tecnologias desenvolvidas pela Microsoft, ou optar pelas soluções oferecidas pela Sun Microsystems e seus aliados através da linguagem Java. Analisando as opções, vemos que embora as tecnologias desenvolvidas pela Microsoft possuam suporte à programação de rede e integração com a Web, elas estão integradas apenas com as plataformas Windows 9x e NT. No caso da Sun [Sun Microsystems, 2000], ela oferece uma solução independente de plataforma, que atende bem nossas necessidades (requisitos). Além da portabilidade, essa tecnologias possuem outras características importantes para o desenvolvimento de aplicações distribuídas: interface para acesso a bancos de dados, capacidade de multi-linhas de execução, suporte à programação em rede e integração com a Web através de tecnologias como *Applets*, *Servlets* e JSP (*Java Server Pages*).

O padrão Java vem despontando como uma importante tecnologia para desenvolvimento de aplicações avançadas, permitindo a utilização de uma abordagem baseada em componentes de software. Componentes de software representam um importante passo no sentido de sistematizar a produção de software, ao prover reusabilidade num alto nível de abstração e, inclusive, através da utilização apropriada de técnicas de orientação a objetos. Padrões como CORBA (*Common Object Request Broker Architecture*) [Siegel, 1996] e COM (*Component Object Model*) [Rofail & Shohoud,

1999], permitem conectar componentes construídos em diferentes linguagens e plataformas, e APIs como *JavaBeans* [Sun Microsystems, 2000] permitem desenvolver componentes de acordo com as características de determinado tipo de linguagem de programação. Ferramentas visuais também podem ajudar o programador a "compor" aplicações através da conexão e configuração dos componentes.

Todas essas características, e outras que deixamos de citar, permitem o desenvolvimento de arquiteturas altamente flexíveis, facilitando a adição de novas funcionalidades e diminuindo o tempo e os custos envolvidos no desenvolvimento. No restante dessa seção, buscaremos descrever como parte desse conjunto de novas e avançadas tecnologias pode ser empregada para atender os requisitos da nossa solução.

3.3.1 Arquitetura baseada em Web

Em primeiro lugar, analisaremos a arquitetura sobre a qual será estruturada a aplicação, mantendo a concordância com os requisitos **NF4** e **NF5**. Como vimos na seção 2.7, as aplicações baseadas em Web são estruturadas em duas ou mais camadas, de acordo com as características da solução. As camadas servem para separar aspectos distintos, cada um devendo ser tratado como um problema a parte. Dessa forma, a aplicação terá uma arquitetura seguindo um modelo em três camadas para abordar os problemas relacionados com a aquisição, o tratamento e a visualização da informação de gerência (figura 3.5).

Figura 3.5 – Aplicação de Gerência baseada em Web

A primeira camada, de acordo com o requisito **NF6,** consiste dos agentes SNMP contidos nos equipamentos gerenciados (roteadores, hospedeiros, etc.). A segunda camada, de acordo com o requisito **NF7**, é formada pela estação de gerência, que possui duas funcionalidades principais. A primeira funcionalidade é de obter dados da primeira camada via protocolo SNMP, em concordância com o requisito **NF1,** e armazenar as informações sobre os equipamentos. A segunda funcionalidade é converter os dados obtidos para formato HTML (*Hyper Text Markup Language*), disponibilizando as informações através de uma interface Web na terceira camada da aplicação, respeitando o requisito **NF8**.

3.3.2 Aquisição da Informação de Gerência

Nesta seção, descreveremos uma arquitetura base para o núcleo da aplicação, buscando soluções para os problemas relacionados com a aquisição da informação de gerência, onde os principais pontos abordados são: a monitoração dos equipamentos, a manipulação da informação de gerência e a indicação do status dos equipamentos.

Para lidar com estes problemas, aplicaremos uma abordagem baseada em componentes. A abordagem baseada em componentes é reconhecida atualmente como a melhor forma para se construir e lidar com a complexidade crescente dos sistemas, prometendo reduzir o tempo e o esforço empregado na construção de aplicações.

Nessa abordagem, componentes são implementados através da **orientação a objetos** (OO). Dessa forma, uma aplicação construída através de componentes é um conjunto classes e interfaces que cooperam entre si, onde cada componente implementa uma determinada funcionalidade da aplicação. Da cooperação entre os componentes utilizados, obtém-se a funcionalidade geral da aplicação. No nosso caso, identificamos cinco funcionalidades básicas que podem ser distribuídas entre os componentes da aplicação:

- **Monitor:** responsável pela aquisição da informação de gerência nos agentes SNMP espalhados na rede. Disponibiliza também a informação de gerência para os outros módulos da aplicação (requisito **F1**);
- **Gerador de Eventos -** responsável por gerar os eventos interessantes a partir da informação de gerência coletada na rede (requisito **F5**);
- **Logador:** responsável por armazenar os eventos gerados e a informação de gerência coletada na rede, provendo acesso às estatísticas históricas sobre o comportamento dos equipamentos gerenciados (requisito **F3**) ;

- **Mailer** - responsável pelo envio de mensagens eletrônicas quando algum problema é identificado na rede (requisito **F8**).
- **Status –** Responsável por identificar e disponibilizar o estado de funcionamento dos dispositivos gerenciados (requisito **F5** e **F13**);

Pelas restrições tecnológicas discutidas no inicio da seção 3.3, escolhemos projetar a solução inteiramente utilizando componentes JavaBeans. Um JavaBean é um componente reutilizável de software cujos métodos e propriedades podem ser reconhecidos em tempo de execução e manipulados dinamicamente pelas aplicações. Esses componentes tanto podem ser encaixados para construção de aplicações sofisticadas como utilizados para a criação de outros JavaBeans dentro de ambientes de edição visual.

Figura 3.6 – Núcleo da Aplicação de Gerência

Devido à natureza da aplicação estar voltada para a manipulação dinâmica de dados, escolhemos conectar os módulos da aplicação através de uma arquitetura Infobus como mostra a figura 3.6. Infobus é uma tecnologia para compartilhamento dinâmico de dados que permite equipar os componentes JavaBeans para se comunicarem entre si de modo estruturado [Colan, 1998]. O Infobus funciona com um repositório onde os componentes

podem se agregar para gerar ou obter dados. Dentro dessa arquitetura, os componentes que se agregam ao Infobus para gerar dados são chamados de produtores, enquanto os que obtém dados do Infobus são chamados consumidores.

A aplicação também utiliza esta tecnologia para obter uma melhor modularização, de modo que as tarefas sejam bem distribuídas entre seus componentes. Este tipo de arquitetura de comunicação entre componentes também se apresenta muito flexível, facilitando a inclusão de novos módulos para estender as funcionalidades da aplicação.

A figura 3.6 mostra uma perspectiva para arquitetura da aplicação através do fluxo de dados entre seus componentes principais. Neste esquema, um componente Monitor é encarregado de coletar periodicamente informações de gerência junto aos agentes SNMP na rede e depositá-las no Infobus de Gerência. A cada nova informação produzida no Infobus de Gerência, uma notificação de mudança é enviada para os componentes consumidores à ele conectados. Dessa maneira, o componente Logador, o componente Status e o componente Gerador de Eventos podem acessar e tratar esta informação: o Logador acessa e armazena o conteúdo da informação em arquivos de dados; o Status identifica e disponibiliza o status dos dispositivos para os outros módulos da aplicação; e o Gerador de Eventos verifica ocorrências de eventos interessantes na rede.

Caso alguma ocorrência interessante seja identificada, uma informação sobre o evento é produzida no Infobus de Eventos. No Infobus de Eventos, essa informação também pode ser acessada e armazenada em arquivos de dados, bem como ocasionar o envio de e-mails para o encarregado pela rede gerenciada.

3.3.3 Tratamento e Visualização da Informação de Gerência

Os problemas relacionados com o tratamento e a visualização da informação de gerência são: a geração de imagens e construção das páginas HTML sobre a rede. Vamos agora falar sobre cada um desses problemas e as suas possíveis soluções. É importante ressaltarmos que optamos pela construção de uma aplicação baseada em Web, de acordo com o requisito **NF4**. Portanto, as soluções listadas satisfazem esta escolha, levando em consideração os requisitos levantados anteriormente.

Problema 1: Geração de imagens.

As imagens geradas pela aplicação devem apresentar a rede gerenciada, o estado de funcionamento dos equipamentos gerenciados e gráficos mostrando a modificação de suas

variáveis no tempo, de acordo com os requisitos **F12**, **F13** e **F17**. Podemos aplicar para este problema duas soluções distintas:

Solução 1: Utilizar uma API de applets Java para construir imagens. Nesta solução, são passados para o browser do cliente as informações necessárias e o applet se encarrega de gerar as imagens e apresentá-las na interface;

Solução 2: Utilizar uma API para gerar imagens no formato GIF a partir de um programa Java (um *servlet*) sendo executado no servidor. Assim, por exemplo, podemos construir um programa Java que, a partir dos dados coletados de um elemento, gere um gráfico e depois transforme este gráfico em uma imagem GIF. A resposta ao cliente seria uma página HTML que tenha uma referência a essa imagem.

Comentários e escolha:

A solução 1, apesar de proporcionar geração de gráficos visualmente sofisticados, parece inadequada para solução deste problema por três razões. Em primeiro lugar, a carga de applets Java pode ser lenta para atender os requisitos de desempenho **F9**, **F10**, **F11**. Os applets Java, como já mencionado anteriormente, compõem uma solução *client-side*, isto é, existe computação no lado do cliente. Em segundo lugar, e muito mais importante, a utilização de applets dificultaria a impressão das páginas, pois os browsers não imprimem o que é exibido pelo applet. Finalmente, a solução dificultaria o envio de um relatório via e-mail (requisito **F9**), haja vista que os browsers não exportam a informação da janela do applet para a área de transferência, e esta seria a forma mais simples de enviar o relatório via e-mail. O mesmo problema ocorre caso o browser permita enviar a página HTML em si via e-mail: o conteúdo da janela do applet não é enviado. A solução 2 foi a escolhida, porém não está claro que esta solução apresentará um melhor desempenho que a solução 1.

Estimando o tamanho do applet que construiria as imagens em 30Kbytes e o tamanho de cada imagem GIF em aproximadamente 4Kbytes, se uma página contiver 7 imagens então o applet e as imagens terão aproximadamente o mesmo tamanho. Dessa forma, o que mais tomará tempo serão as conexões estabelecidas entre o cliente e o servidor.

O número de conexões estabelecidas entre o servidor e o cliente na solução 1 é quase igual ao número de conexões estabelecidas entre o servidor e o cliente na solução 2. Na solução 1 as seguintes conexões entre o servidor e o browser cliente serão estabelecidas: uma para carregar o arquivo HTML, outra para carregar o applet e para cada gráfico que tiver que ser gerado deve-se abrir uma conexão para receber do servidor os dados que

serão usados para a construção dos gráficos. Na solução 2 teremos apenas uma conexão a menos: será aberta uma conexão para carregar a página HTML e uma conexão para trazer cada imagem GIF do servidor. Pelo exposto verificamos que não podemos afirmar que a solução 2 nos oferecerá um melhor desempenho. Na realidade, o que nos motivou a escolher a solução 2 é fato de que satisfaz mais requisitos que as soluções 1. A solução 2 nos permite imprimir as páginas e nos oferece um alto grau de sofisticação gráfica.

Problema 2: geração de páginas HTML da rede.

O solução deve ser capaz de mostrar informações sobre a rede e disponibilizá-las na Web. Portanto, precisamos encontrar um meio de gerar os arquivos HTML (os arquivos que são apresentados ao cliente através de um browser) que farão referência às imagens geradas no problema 1. É importante observar que as páginas devem ser construídas dinamicamente, uma vez que as informações na interface devem refletir o estado atual da rede. Por exemplo, o estado de funcionamento dos equipamentos e os gráficos gerados podem se alterar sempre que uma página for atualizada. Diante desta restrição, encontramos três possibilidades de construção de páginas HTML dinâmicas:

Solução 1: Utilizar JSP (*Java Server Pages*);

Solução 2: Utilizar ASP (*Active Server Pages*);

Solução 3: Utilizar CGI (*Common Gateway Interface*).

Comentários e escolha:

A solução 1 foi a escolhida. A tecnologia ASP foi excluída por falta de portabilidade: é uma tecnologia da Microsoft e só funciona se o servidor Web usado for o *Internet Information Server* (da Microsoft), não atendendo completamente o requisito **NF7**.

A tecnologia CGI, abordada na solução 3, poderia perfeitamente ser usada para solucionar o problema de geração dinâmica de páginas HTML. O maior problema desta tecnologia é o baixo desempenho em relação às outras tecnologias citadas (JSP e ASP): a tecnologia CGI cria um novo processo no servidor para atender a cada pedido do cliente, enquanto as outras tecnologias citadas se utilizam de linhas paralelas de execução para atender aos diversos pedidos dos clientes. No entanto, no nosso caso, este baixo desempenho relativo não seria problemático, pois não teríamos muitos clientes simultâneos acessando a aplicação. O que nos levou a escolher a tecnologia JSP, foi o nosso desejo de utilizar arquiteturas de aplicação mais geralmente aplicáveis, isto é, aplicáveis a uma vasta

gama de situações. Devido à portabilidade e ao bom desempenho, a tecnologia JSP poderá sempre ser aplicada para gerar páginas HTML dinâmicas.

Ao utilizar esta arquitetura de aplicação iremos, também, isolar a *Business Logic* da *Presentation Logic* para facilitar a manutenção do sistema, uma vez que o analista/programador responsável por definir novas páginas não está apto a construir complexas estruturas. *A Business Logic* é a funcionalidade que queremos dar ao sistema e a *Presentation Logic* é a forma de apresentação (interface) do sistema.

Esta solução possibilita que pessoas diferentes, com perfis profissionais diferentes, possam tratar de cada lado do sistema: um designer pode construir as páginas Web (*Presentation Logic*) e um profissional de redes com nível de programador pode ficar responsável pela funcionalidade do sistema (*Business Logic*), embutida em componentes apropriados. Conseguiremos este isolamento encapsulando a interface necessária para construção da interface da aplicação numa classe Java que possui métodos apropriados para buscar e expor as informações sobre os equipamentos gerenciados. A classe que constrói as páginas HTML será então um JavaBean, isto é, um componente que encapsula todo o código complexo necessário, por exemplo, para informar o estado dos equipamentos gerenciados, ou pelo menos acessar outra classe que encapsule esta verificação.

Na figura 3.7, encontramos um esquema gráfico da arquitetura geral do sistema em desenvolvimento. O gerente de rede terá acesso à aplicação através de um browser, acessando o servidor Web da estação de gerência. Neste servidor, ele deverá encontrar páginas estáticas e páginas construídas dinamicamente. As páginas construídas dinamicamente têm (pela escolha anterior) terminação JSP, e seu conteúdo é formado de trechos de HTML junto com um código Java que deve ser executado no servidor.

Este código dá o dinamismo à página, pois o resultado de sua execução pode ser diferente a cada acesso. O conteúdo de uma página JSP gera o que chamamos de servlet JSP, que é o trecho de código executado no servidor. Esse servlet carregará componentes que possuem métodos adequados para acessar e expor as informações na interface Web. Esses componentes são basicamente de três tipos: componentes para acesso à informação de configuração da aplicação (nome da rede, modelo dos equipamentos, etc.), componentes para acesso às estatísticas dos equipamentos (estado dos equipamentos, valores das variáveis, etc.) e componentes gráficos (gráficos estatísticos, imagens da rede, etc.).

É importante observar que os componentes que constróem as páginas Web são carregados em uma JVM[2] (*Java Virtual Machine*) diferente da que são carregados os componentes que monitoram e disponibilizam as informações sobre os equipamentos gerenciados. Neste caso, a forma mais simples de possibilitar a troca de mensagens entre esses componentes é através de RMI (*Remote Method Invocation*). RMI permite que um componente rodando em qualquer JVM invoque métodos de outro componente rodando em outra JVM, provendo uma forma padronizada de efetuar a comunicação remota entre programas escritos na linguagem de programação Java.

Figura 3.7 – Arquitetura geral da solução

[2] Toda classe Java, após compilada, precisa ser executada (interpretada) em uma JVM, sendo esta uma forma padronizada de garantir a portabilidade para as aplicações desenvolvidas utilizando essa tecnologia.

3.3.5 Componentes da Interface Web

Descreveremos a seguir os objetos criados pela aplicação que são visualizados pelo usuário através da interface Web, de acordo com os requisitos **F12**, **F13**, **F14**, **F15 e F17**.

Imagens da rede: arquivos GIFs construídos dinamicamente para mostrar de forma gráfica a rede, os equipamentos gerenciados e as estatísticas sobre o comportamento de variáveis, podendo ser de dois tipos diferentes:

(i) **Gráficos Estatísticos** – tipo de representação abscissa versus ordenada, mostrando o comportamento das variáveis dos equipamentos no tempo (24h), onde o gráfico se expande para direita descrevendo uma ou mais linhas (figura 3.8);

Figura 3.8 – Exemplo de gráfico estatístico sobre variáveis MIB

(ii) **Mapas da rede** - mostram graficamente a rede, indicando a situação dos equipamentos gerenciados pelo uso de pontos coloridos: um ponto verde sobre um equipamento ou canal de comunicação significa um estado operacional correto; pontos pretos, amarelos, laranjas e vermelhos indicam problemas de diferentes níveis críticos (figura 3.9).

***Frames* HTML:** arquivo que define uma estrutura em janelas (quantas e qual sua distribuição em tela) para um documento da Web, onde para cada janela está associada uma URL (*Uniform Resource Locator*).

Figura 3.9 - Exemplo de um Mapa da Rede

Páginas JSP: arquivos que definem os vários componentes presentes nos documentos Web (texto, imagens, formulários, *links*, etc.). Os seguintes grupos de páginas devem ser geradas pela ferramenta:

(i) **Páginas da Rede** - apresentam um visão da rede gerenciada através da sua topologia e do estado operacional dos dispositivos existentes, distribuído em vários mapas organizados numa estrutura em grafo (figura 3.10);

(ii) **Páginas dos Equipamentos** - permitem acessar os atributos sobre um equipamento gerenciado (descrição, endereço de rede, localização, etc.) e os gráficos estatísticos sobre as variáveis gerenciadas, sendo referenciadas através dos *hiperlinks* presentes nas **páginas da rede**;

(iii) **Páginas de Gráficos** - permitem visualizar gráficos com estatísticas atuais ou históricas sobre as variáveis gerenciadas, sendo referenciadas através dos *hiperlinks* ou formulários presentes nas **páginas dos equipamentos**;

Figura 3.10 – Esquema dos mapas da rede

3.3.6 Configuração da Aplicação

Abordamos nesta seção os problemas relativos à configuração da aplicação de gerência, de acordo com os requisitos de **F22** a **F28**. Os principais pontos abordados são: a configuração dos equipamentos, dos gráficos e dos mapas da rede. Nesta versão, não cumpriremos nenhum requisito de autotopologia ou reconhecimento automático dos dispositivos, ficando a cargo do usuário definir todas as informações necessárias para que a aplicação possa fazer seu trabalho. Dessa forma, a primeira questão que surge é como o usuário pode configurar os vários componentes da aplicação.

Devido às escolhas feitas anteriormente, decidimos utilizar uma API Java que será utilizada pela aplicação para acessar informações de configuração que serão especificadas utilizando uma linguagem de simples sintaxe, dentro de um arquivo do tipo texto, sem exigir do usuário conhecimento de nenhuma linguagem de programação. Através desta API, deverá ser possível definir e configurar os componentes JavaBeans que serão utilizados para a configuração da aplicação. Esta abordagem para a configuração de aplicações baseadas em componentes vem se tornando bastante comum, onde XML é

utilizada para especificar a linguagem de configuração dos componentes. XML (*Extensible Markup Language*) é uma linguagem de marcação baseada em texto que está se tornando rapidamente o padrão utilizado para a troca de dados na Web. Existem vários modos de se fazer uso de XML [Armstrong, 1999]. Na nossa abordagem, utilizamos XML para configuração dos componentes utilizados pelos módulos da aplicação e dos componentes gráficos na Interface Web do usuário. Para exemplificar o uso de XML, temos na figura 3.11 a configuração de um equipamento de rede gerenciado pela aplicação, onde o componente configurado (`NetEquipment`) e suas propriedades (nome, descrição do hardware, endereço de rede, etc.) são especificados por *tags*.

```
<NetEquipment>
   <name>sw8273</name>
   <hardwareDescription>RouterSwitch IBM 8273 </hardwareDescription>
   <functionalDescription>roteador do POP-PB</functionalDescription>
   <criticality>9</criticality>
   <host>sw1.pop-pb.rnp.br</host>
   <port>161</port>
   <protocol>0</protocol>
   <retries>0</retries>
   <timeout>4000</timeout>
   <pollInterval>5</pollInterval>
   <pollsToSave>3</pollsToSave>
   <community>RNP</community>
   <writeCommunity>private</writeCommunity>
   <instanceIndex>0</instanceIndex>
   <statusOID>.webmngr.ping.PingUp</statusOID>
   <allGraphics>
         <variableGraphic>
               <title>Tempo de Ping</title>
               <oid>.webmngr.ping.PingTime</oid>
               <label>ms</label>
               <limit>1000</limit>
         </variableGraphic>
   </allGraphics>
</NetEquipment>
```

Figura 3.11 – Exemplo de configuração do equipamento usando XML.

Observe que, para cada propriedade, temos uma tag <*propriedade*> informando o valor configurado. Por exemplo, o nome do equipamento está representado no XML como `<name>sw8273</name>`, com valor igual a "sw8273". Se o componente contém uma propriedade representada por uma lista ou um conjunto de informações, representamos esta propriedade da mesma forma, repetindo tantas vezes quanto queremos que ela seja adicionada aos componentes configurados. É importante destacar que uma propriedade de um componente pode ser outro componente. Por exemplo, `VariableGraphic` representa um gráfico estatístico que pode aparecer na página Web de um equipamento.

Usando XML, o usuário poderá fazer as configurações através de editores de texto ou também editores gráficos XML disponíveis no mercado, a exemplo do Xeena da IBM. Certos editores visuais disponíveis atualmente permitem que com base no arquivo texto seja exibida uma estrutura de árvore. Tal formato possibilita uma melhor visualização dos componentes da rede e facilita o processo de edição de suas propriedades.

3.3.7 Síntese por Componente Apresentado

Finalizamos este capítulo mostrando um resumo da arquitetura proposta através dos componentes principais da aplicação que foram discutidos durante o projeto de alto-nível. Estes componentes especificam a arquitetura externa da solução, e servirão de base para o projeto detalhado e implementação das classes necessárias à aplicação de gerência.

Síntese por componente:

Browser: ferramenta que o usuário utilizará como interface para acessar as páginas sobre a rede e os equipamentos gerenciados ;

Página JSP: páginas que contêm código HTML e código em Java. Quando um usuário solicita uma página JSP um servlet JSP é executado no servidor. Este servlet executará um código em Java que tem como função buscar os dados que deverão ser exibidos na página Web.

Servlet JSP: programa em Java, atrelado a alguma página JSP que é executado quando esta página é requisitada. O servlet é gerado automaticamente no primeiro acesso à página JSP correspondente.

Módulo Monitor: conjunto de componentes responsáveis pela aquisição da informação de gerência, bem como a disponibilização da informação coletada para os outros módulos da aplicação através do Infobus;

Módulo Gerador de Eventos: conjunto de componentes responsáveis por identificar os eventos interessantes na rede a partir da informação de gerência obtida;

Módulo Infobus: conjunto de componentes responsáveis pelo compartilhamento dinâmico dos dados de gerência e dos eventos importantes identificados a partir da informação de gerência para todos os outros módulos da aplicação;

Módulo Logador: conjunto de componentes responsáveis por armazenar a informação disponibilizada no Infobus, provendo acesso às estatísticas históricas sobre o comportamento dos equipamentos gerenciados;

Módulo Mailer : conjunto de componentes responsáveis pelo envio de e-mails, quando algum problema é identificado na rede;

Módulo Servlet: conjunto de componentes que serão instanciados pelo servlet para gerar as informações que serão apresentados na página sobre os dispositivos. No momento é interessante saber que teremos, pelo menos, três tipos de componentes auxiliares: componentes para acesso à informação de configuração, componentes para acesso às estatísticas dos equipamentos e componentes gráficos.

Módulo Config: conjunto de componentes utilizados para configuração da aplicação de gerência (dispositivos gerenciados, gráficos, etc.).

Capítulo 4

Projeto Detalhado e Implementação

Neste capítulo, levantamos detalhes do projeto e implementação da solução. Na seção 4.1, apresentamos sua organização geral, mostrando o processo de desenvolvimento utilizado, o planejamento da construção e os pacotes desenvolvidos. Na seção 4.2, apresentamos as APIs utilizadas que foram fornecidas por outros desenvolvedores. Na seção 4.3, apresentamos as classes da aplicação, ressaltando os detalhes importantes de implementação. Na seção 4.4, mostramos os procedimentos de testes realizados. Na seção 4.5, finalizamos o capítulo mostrando a implementação da interface da aplicação.

4.1 Organização e Construção do Projeto

O desenvolvimento desta aplicação se baseia nas etapas básicas de qualquer processo de desenvolvimento de software [Rumbaugh *et al*., 1999]: análise do domínio do problema, projeto da solução, implementação e testes. As técnicas empregadas na engenharia de software atual apontam que um bom processo de desenvolvimento deve ser:

- **Iterativo -** possui várias iterações no tempo, onde cada iteração trata de uma parte da funcionalidade ou característica atribuída ao software;
- **Incremental -** ao final de cada iteração, devemos ter sempre novas versões incrementadas em funcionalidade e integradas em relação à versão anterior.

Dessa forma, o desenvolvimento desta aplicação emprega um processo iterativo incremental baseado no *Rational Unified Process* (RUP) [Rumbaugh *et al*., 1999].

No capítulo anterior, realizamos o projeto da solução, propondo uma arquitetura base para o cumprimento dos requisitos levantados durante a análise do domínio do problema. Os componentes dessa arquitetura foram organizados em pacotes de software de acordo com suas funcionalidades e implementados através das seguintes iterações:

Iteração	**Módulo**	**Pacote**	**Responsabilidades**
I	Iniciador	`webmngr`	Inicia a aplicação de gerência, instanciando e configurando seus componentes principais
I	Monitor	`webmngr.monitor`	Monitoração dos equipamentos gerenciados utilizando a arquitetura padrão da Internet
I	Util	`webmngr.util`	Utilidades gerais da aplicação
I	Logador	`webmngr.log`	Armazenamento das informações produzidas no Infobus, provendo acesso às estatísticas históricas
II	Config	`webmngr.config`	Configuração da aplicação através de componentes de software e XML
III	Status	`webmngr.status`	Verificação e indicação do estado de funcionamento dos equipamentos monitorados pela aplicação
III	Servlet	`webmngr.servlet`	Apresentação das informações sobre os equipamentos na interface Web da aplicação (atributos, gráficos estatísticos e status dos dispositivos)
IV	Eventos	`webmngr.event`	Geração dos eventos interessantes a partir da informação de gerência coletada na rede
IV	Mail	`webmngr.mail`	Envio de e-mails para os responsável pela rede gerenciada

Tabela 4.1 – Iterações realizadas e os pacotes desenvolvidos

Além das classes implementadas, a aplicação utiliza outros arquivos e recursos organizados numa estrutura de diretórios, da seguinte forma:

- `/webmngr`: diretório principal da aplicação onde podemos encontrar as classes, arquivos e demais recursos utilizados e organizados em subdiretórios;
- `/webmngr/lib`: bibliotecas de classes e métodos nativos utilizados;
- `/webmngr/mibs`: arquivos de definição, contendo os módulos MIB carregados pela aplicação;
- `/webmngr/source`: código fonte das classes implementadas;
- `/webmngr/data`: arquivos contendo os dados coletados na rede;
- `/webmngr/config`: arquivos de configuração (apêndices A e B);
- `/webmngr/bin`: arquivos executáveis da aplicação (Win9x/NT);
- `/webmngr/docs`: javadocs, especificando as APIs desenvolvidas;
- `/webmngr/logs`: arquivos de log gerados pela aplicação;
- `/webmngr/web`: diretório web da aplicação, que reúne os recursos necessários para construção das páginas apresentadas na interface do usuário (figuras, modelos JSP, páginas HTML, etc.).

4.2 APIs Utilizadas

Nesta seção, descrevemos as APIs fornecidas por outros desenvolvedores que foram utilizadas para implementação desta aplicação. Sob a forma de projetos de pesquisa, algumas dessas APIs foram desenvolvidas em cooperação com alunos de Iniciação Científica e do Grupo PET (Programa Especial de Treinamento) do Departamento de Sistema e Computação da UFPB.

4.2.1 API Infobus

A arquitetura definida para aplicação foi baseada na API InfoBus, uma especificação pública desenvolvida pela Lotus Corporation e Sun Microsystems [InfoBus, 2000]. Esta API define um conjunto de interfaces que devem ser implementadas para permitir a troca estruturada de dados entre os componentes de uma aplicação. Nesta arquitetura, encontramos os seguintes elementos principais:

- **Produtores** – componentes que geram dados no InfoBus;
- **Consumidores** - componentes que acessam os dados produzidos no InfoBus;
- **Itens de dados -** encapsulam as informações que fluem através do InfoBus.

Um produtor sempre anuncia a disponibilidade dos itens por ele produzidos para os outros membros agregados ao InfoBus. Um consumidor pode pedir um item de dados em reação ao anúncio de um produtor sobre sua disponibilidade, ou requisitar uma informação particular através do nome do item ou de uma referência ao seu produtor. Quando isto acontece, o produtor cria uma instância do item de dado requisitado e provê a informação para o consumidor que efetuou o pedido. Quando um consumidor recebe um item de dado de um produtor, ele pode efetuar modificações no item ou se registrar para ser notificado sobre todas as mudanças realizadas. O produtor é quem descobre as modificações e as anuncia para todos os consumidores interessados. Essas mudanças também podem ser ocasionadas por fontes de dados externas, relacionadas aos produtores.

A API InfoBus especifica várias interfaces padrões, provendo um protocolo para a transferência de dados entre os produtores e consumidores. Dentre elas, destacamos a seguir as interfaces que foram utilizadas na implementação dos componentes da solução:

- `InfoBusMember` e `InfoBusBean` - interfaces implementadas pelos componentes que acessam o InfoBus. O propósito principal é prover métodos que permitem manipular a instancia do InfoBus ao qual o componente esta agregado bem como suas propriedades;
- `InfoBusDataProducer` - interface implementada pelos produtores, provendo métodos para receber eventos que indicam pedidos dos consumidores pelos itens de dados oferecidos;
- `InfoBusDataConsumer` - interface implementada pelos consumidores, provendo métodos para receber eventos que indicam a disponibilidade e revogação dos itens de dados oferecidos pelos produtores;
- `DataItemChangeManager` - interface implementada nos itens de dados para os quais se deseja disponibilizar notificações de mudanças. Seu propósito principal é prover métodos que permitem aos consumidores se registrarem para receber notificações de mudanças pelos consumidos;

- `DataItemChangeListener` – interface implementada pelos consumidores, provendo métodos para permitir a notificação por parte dos produtores sobre mudanças que acontecem nos itens de dados;
- `DataItem` – interface implementada pelos itens de dados, provendo métodos que permitem obter o tipo do dado encapsulado, referências para seus produtores ou o valor associado a uma determinada propriedade;
- `ImmediateAccess` – interface implementada pelos itens de dados, fornecendo métodos para manipular os dados encapsulados. Pode-se recuperar os dados na forma de strings ou objetos, ou lançar exceções caso o item seja apenas de leitura.

4.2.2 API AdventNet SNMP

Para oferecer suporte à arquitetura de gerência SNMP, utilizamos uma API fornecida pela AdventNet, Inc. [AdventeNet, 2000]. Desta API, os seguintes pacotes foram utilizados para implementação da aplicação de gerência:

- `com.adventnet.snmp.snmp2` - fornece as classes utilizadas para efetuar as operações de gerência nos agentes SNMP da rede, provendo métodos para configuração dos parâmetros de comunicação utilizados;
- `com.adventnet.snmp.mibs` - provê suporte para MIB/SMI, oferecendo classes que permitem interpretar as informação sobre a organização da informação e sobre os tipos de dados disponíveis num agente SNMP.

4.2.3 API JConfig

Desenvolvida pelos alunos de graduação Rodrigo Rebouças de Almeida e Ayla Débora Dantas de Souza (DSC/UFPB), a API JConfig [Almeida e Souza, 2000a] é um pacote que permite instanciar componentes (JavaBeans) utilizando XML. Esta API provê classes e métodos que permitem interpretar um arquivo XML e retornar uma lista de componentes. Estes componentes são especificados através de uma linguagem semelhante a HTML, onde os *tags* correspondem às propriedades dos componentes.

Aplicações geralmente utilizam componentes para realizar determinadas configurações de software. Utilizando esta API, as configurações podem ser realizadas através de um ou mais arquivos XML fornecidos pelo usuário.

4.2.4 API Map

Desenvolvida pelos alunos de graduação Rodrigo Rebouças de Almeida e Ayla Débora Dantas de Souza (DSC/UFPB), a API Map [Almeida e Souza, 2000b] fornece um conjunto de classes que permitem a construção de sistemas de navegação Web para redes de computadores gerenciadas. Projetada para atender os requisitos de interface levantados na seção 3.2.1, e especificados na seção 3.3.5, esta API aproveita parte das funcionalidades implementadas neste trabalho para mostrar o estado de funcionamento da rede, oferecendo atualização das informações apresentadas em um intervalo máximo de 10 segundos. Estas informações são obtidas via RMI (*Remote Method Invocation*), consultando remotamente os componentes da aplicação.

Nesta API, a rede gerenciada é representada através de um conjunto de mapas que mostram imagens dos elementos gerenciados, linhas, descrições textuais, hiperlinks, e submapas. Utilizando a API JConfig, o sistema oferece uma completa configuração através de arquivos XML que descrevem os mapas da rede e seus elementos relacionados (no apêndice B, mostramos um exemplo completo de configuração dos mapas). É importante salientar que a integração deste sistema com a aplicação foi facilmente obtida devido à utilização de uma interface Web, comum para os dois sistemas desenvolvidos.

4.2.5 API Graphic

Desenvolvida pela aluna de graduação Raquel Vigolvino Lopes (DSC/UFPB), a API Graphic [Lopes, 2000] fornece um pacote de classes que permitem a construção e geração de imagens sobre gráficos estatísticos. Projetada para atender os requisitos de interface levantados na seção 3.2.1, e especificados na seção 3.3.5, esta API foi utilizada para gerar imagens contendo gráficos estatíticos sobre o comportamento das variáveis monitoradas nos dispositivos gerenciados, que são referenciadas nas páginas Web da aplicação.

4.2.6 API JavaMail

Para oferecer suporte ao envio de e-mails pela aplicação de gerência, utilizamos a API JavaMail fornecida pela Sun Microsystems [Sun Microsystems, 2000]. Esse pacote fornece um conjunto de classes abstratas que definem os objetos de um sistema de correio eletrônico. Essa API fornece classes como `Message`, `Store` e `Transport`, que podem ser estendidas para adicionar novos protocolos ou adicionar outras funcionalidades quando necessário. Em adição, a API também oferece subclasses concretas das classes abstratas, incluindo `MimeMessage` e `MimeBordPart`, prontas para serem usadas em aplicações.

4.3 Pacotes Desenvolvidos

Na figura 4.1, temos uma visão geral da aplicação através de um diagrama que mostra as dependências entre os pacotes da aplicação. Nos tópicos seguintes, apresentaremos as classes desenvolvidas em cada pacote da aplicação, ressaltando os detalhes de implementação importantes. No Apêndice C, podem ser encontrados também os diagramas de classes em cada pacote da aplicação.

Figura 4.1 – Diagrama de pacotes

4.3.1 webmngr

Este pacote contém apenas uma classe executável chamada `WebManager`, possuindo um método estático `main()` responsável por instanciar os módulos da aplicação (figura 4.1). Esta classe deve ser executada da seguinte forma:

```
java WebManager [ opções ] [ argumentos_restantes ]
```

onde os seguintes argumentos podem ser passados pela linha de comando:

- ***verbose***: mostra mensagens sobre as atividades executadas na saída padrão;
- ***loadmibs***: compila os arquivos MIB. Novos módulos podem ser adicionados no diretório de arquivos MIB para serem carregados pela aplicação;
- ***mibsdir***: especifica o diretório contendo os arquivos MIB;
- ***configdir***: especifica o diretório contendo os arquivos de configuração;
- ***datadir***: especifica o diretório contendo os arquivos de dados;
- ***logsdir***: especifica o diretório contendo os arquivos de log;
- ***alarmsdir***: especifica o diretório contendo os arquivos de alarmes;
- ***help***: mostra o help da aplicação na saída padrão;
- ***argumentos restantes***: nomes dos equipamentos a monitorar.

Este método realiza a configuração inicial a partir dos atributos passados pela linha de comando e registra no servidor RMI os componentes remotos da aplicação. Na figura 4.1, vemos que a classe `Configuration` obedece o padrão GoF (*Gang of Four*) Singleton. Esse padrão permite que apenas uma única instância dessa classe possa existir, fornecendo um ponto comum de acesso à configuração para todos os outros componentes.

Figura 4.2 – Diagrama de colaboração I

4.3.2 Pacote webmngr.config

Este pacote prove um conjunto de componentes que são utilizados pela aplicação para fins de configuração. Estes componentes podem ser especificados nos arquivos XML da aplicação (apêndice A). Neste pacote, as seguintes classes foram implementadas:

- `Configuration` - componente que encapsula as configurações gerais da aplicação (nome dos diretórios, MIBs e outros objetos compartilhados, etc.);

- `NetElement` - representa elementos gerenciados em uma rede TCP/IP. Estes elementos tanto podem ser dispositivos de hardware como componentes de software. A Implementação desta classe é baseada na arquitetura de gerência SNMP, reunindo métodos e atributos comuns a todos os elementos. Pode ser estendida para criação de novas classes de elementos gerenciáveis;

- `NetEquipment` - subclasse de `NetElement` criada representar equipamentos presentes na rede como roteadores, switches, impressoras, computadores hospedeiros, etc.;

- `NetInterface` - subclasse de `NetElement` criada para representar as interfaces de rede de equipamentos como roteadores, switches, impressoras, computadores hospedeiros, etc.;

- `VariableGraphic` - esta componente representa as configurações para os gráficos estatísticos gerados na página Web dos equipamentos.

4.3.3 Pacote webmngr.monitor

Neste pacote possui um conjunto de componentes utilizados para permitir a aquisição da informação de gerência nos agentes SNMP existentes na rede, bem como a disponibilização dessa informação coletada para os outros módulos da aplicação. Essa informação pode ser disponibilizada tanto através do Infobus de gerência, quanto por meio de chamadas a métodos remotos. Neste pacote, as seguintes interfaces e classes foram implementadas para disponibilizar este serviço:

- `RemoteMonitor` - interface que deve ser implementada por objetos que desejam oferecer um serviço de gerenciamento remoto via RMI;

- `SnmpAgentPoller` - componente instanciado para coletar periodicamente informações de gerência sobre um determinado equipamento na rede, utilizando o protocolo de gerência SNMP. As configurações de monitoração e comunicação utilizadas são obtidas através de um objeto do tipo `NetElement`;

- `ResponseEvent` - eventos gerados por um objetos que realizam a monitoração dos equipamentos, e encapsula o resultado de uma monitoração realizada;

- `SnmpPollerException` - exceção utilizada para indicar e descrever uma falha na monitoração SNMP dos equipamentos;

- `PingICMP` - classe que implementa a funcionalidade do comando *ping* através de chamadas a métodos nativos[1].

- `ResponseListener` - interface implementada por classes que desejam receber eventos do tipo `ResponseEvent`, gerados por objetos que coletam periodicamente dados de gerência na rede, como o `SnmpAgentPoller`.

- `SnmpManager` - controla a monitoração dos dispositivos gerenciados, sendo responsável pela disponibilização da informação coletada para os outros módulos da aplicação. Implementa as interfaces `InfoBusMember`, `InfoBusBean`, `InfoBusDataProducer`, `ResponseListener` e `RemoteMonitor`;

- `ManagementDataItem` – classes responsáveis pelo encapsulamento das informações de gerência que fluem através do Infobus de gerência, provendo métodos de acesso aos dados transferidos entre os componentes. Implementa as interfaces `DataItem` e `DataItemChangeManager`;

O componente `SnmpManager` instancia, configura e controla os *pollers* SNMP de acordo com as informações existentes no arquivo XML de configuração da rede (apêndice A). A figura 4.3 mostra uma visão da interação entre os componentes do pacote no intuito de prover acesso as informações sobre os dispositivos gerenciados.

[1] Esta classe foi implementada usando métodos nativos porque Java não permite acessar a funcionalidade do protocolo ICMP, necessário para implementar o comando *ping* do UNIX.

Figura 4.3 – Diagrama de colaboração II

4.3.4 Pacote webmngr.log

Provê componentes para o armazenando em disco das informações que fluem através do Infobus, fornecendo suporte a estatísticas históricas sobre o comportamento dos equipamentos gerenciados. Nesta pacote, a seguinte classe foi implementa:

- `LogMonitor` - componente responsável por realizar o log em arquivos de dados das informações que fluem através do Infobus. As configurações necessárias são obtidas da configuração geral. Implementa as interfaces `InfoBusBean`, `DataItemChangeListener` e `InfoBusDataConsumer`.

4.3.5 Pacote webmngr.status

Este pacote provê classes para verificação e disponibilização do estado de funcionamento dos equipamentos para os módulo remotos da aplicação via RMI. Nesta pacote, a seguintes classes e interfaces foram desenvolvidas:

- `StatusMonitor` – componente responsável por verificar o status dos dispositivos a partir das informações coletadas no Infobus de gerência, disponibilizando essas informações para os outros módulos remotos da aplicação via RMI. Implementa as interfaces `RemoteStatus`, `InfoBusBean`, `DataItemChangeListener` e `InfoBusDataConsumer`.

- `RemoteStatus` - interface que deve ser implementada por objetos que desejam disponibilizar informações de status dos equipamentos remotamente via RMI;

4.3.6 Pacote webmngr.event

Este pacote provê classes para verificação e identificação de eventos interessantes que acontecem na rede. Nesta pacote, a seguintes classes e interfaces foram desenvolvidas:

- `EventDataItem` – classe responsável pelo encapsulamento das informações sobre eventos que fluem através do Infobus de eventos, provendo métodos de acesso aos dados transferidos entre os componentes do Infobus. Implementa as interfaces `DataItem` e `DataItemChangeManager`;
- `EventGenerator` - responsável pela geração de eventos no InfoBus de eventos, analisando as informações geradas no Infobus de gerência. Implementa as interfaces `InfoBusMember`, `InfoBusBean`, `InfoBusDataConsumer` `InfoBusDataProducer` e `DataItemChangeListener`;

4.3.7 Pacote webmngr.info

Este pacote provê componentes para encapsular as informações que circulam através do infobus da aplicação. Todas as classes desse pacote implementam a interface `Infodata`. As seguintes classes foram implementas:

- `Infodata` – interface que prove métodos adequados para acessar e manipular as informações que circulam através do infobus;
- `InfoManagement` - encapsula a informação de gerência recuperada de um determinado elemento da rede;
- `InfoStatus` - encapsula informações de status do equipamento;
- `InfoEvent` – encapsula as informações sobre o evento ocorrido;

4.3.8 Pacote webmngr.mail

Este pacote provê componentes para oferecer suporte ao envio de e-mails pela aplicação de gerência. A seguinte classe foi implementa:

- `Mailer` - responsável pela envio de e-mails para o gerente da rede quando um evento é identificado no Infobus de eventos. As configurações necessárias são obtidas através da configuração geral. Implementa as interfaces `InfoBusBean`, `DataItemChangeListener` e `InfoBusDataConsumer`;

4.3.9 Pacote webmngr.servlet

Este pacote contem um conjunto de classes que permitem acessar as funcionalidades de gerência oferecidas pela solução. Esses componentes foram instanciados através de *servlets* no intuito de possibilitar a construção de uma interface baseada em Web para a aplicação de gerência. Neste pacote, as seguintes classes foram implementadas:

- `RemoteManager` - este componente disponibiliza métodos para acessar remotamente as informações (configuração dos elementos, status, etc) sobre os equipamentos gerenciados pela aplicação;
- `VariableSample` - esta classe dá acesso às amostras diárias dos valores das variáveis extraídas dos arquivos de dados da aplicação;
- `GraphicPlotter` - gera gráficos estatísticos a partir das configurações dos equipamentos e das amostras dos valores das variáveis obtidas;

4.3.10 Pacote webmngr.util

Este pacote provê componentes de funcionalidade geral, utilizadas por todos os pacotes da aplicação. As seguinte classes foram implementas:

- `ParseOptions` - classe utilizada para analisar argumentos passados pelo usuário através da linha de comando pelo usuário, fornecendo métodos para definir e recuperar os argumentos passados, definir textos de ajuda, etc;

4.4 Procedimentos de Testes Realizados

Além de implementarmos as classes definidas no projeto, implementamos simultaneamente outras classes para testar as classes construídas. Após efetuarmos os testes, passamos a ter uma maior confiança naquilo que foi implementado e, quando precisamos realizar alguma modificação em uma classe que já funciona, os testes nos dão a certeza de que a classe ainda está funcionando da forma desejada [Gamma e Beck, 1999].

Durante este trabalho, sentimos a importância de testes de unidade executados de forma automática para diminuir o esforço de codificação e manutenção do código. Para implementação dos testes, utilizamos o *framework* JUnit [Gamma e Beck, 1999], que auxilia na construção e execução de testes de unidades para código escrito em Java. O JUnit define como estruturar os testes e provê as ferramentas para executá-los.

Neste esquema, os testes implementados são subclasses de `TestCase`. Como em Java as classes estão contidas em pacotes, temos que decidir onde colocar as classes de testes. Além dos pacotes desenvolvidos, decidimos criar o pacote `webmngr.tests`, que conterá todos os testes para os métodos públicos das classes construídas para aplicação.

Por exemplo, para testar a classe `webmngr.monitor.SnmpManager`, construímos uma classe `webmngr.tests.monitor.TestSnmpManager`, que estende as funcionalidades de `TestCase`, adicionando os teste que serão realizados.

Figura 4.4 – Testando as classes da aplicação com o JUnit

Os métodos das classes de testes são métodos sem parâmetros, que chamam os métodos das classes testadas e compara o resultado obtido com o resultado esperado no teste. Utilizando JUnit, podemos ainda executar automaticamente todos os testes construídos através da linha de comando ou com ajuda de ferramentas visuais (figura 4.3), facilitando a localização e correção dos erros identificados. Atualmente, 150 testes diferentes foram implementados para as classes da aplicação.

4.5 Interface da Aplicação

Ao terminarmos a codificação e os testes das classes dos pacotes, partimos para a implementação das páginas JSP que servem de interface para a aplicação. O desenvolvimento destas páginas foi simples, uma vez que as classes e métodos que oferecem as funcionalidades necessárias já estavam prontos. Verificamos ao construir as páginas que conseguimos alcançar o objetivo de separar a *presentation logic* da *business logic*. No exemplo abaixo, mostramos o modelo de página JSP utilizado para mostrar a lista de equipamentos gerenciados pela aplicação, e seus respectivos status.

```
<%@ page import="webmngr.servlet.*" %>
<%@ page import="webmngr.status.*" %>
<html><head>
<title> Lista dos Equipamentos Gerenciados </title>
<meta http-equiv="Pragma" content="no-cache"><meta http-equiv="refresh" content="5">
</head>
<body bgcolor="#000000" text="#FFFFFF">
<% RemoteManager webmngr = new RemoteManager(); %>
<% String[] names =  webmngr.getNames(); %>
<b>Lista dos equipamentos gerenciados:</b><BR>
<% for (int i = 0; i < names.length; i++) {
   InfoStatus status = webmngr.getInfoStatus(names[i]);
   if (status.getStatus() == InfoStatus.STATUS_UP)
       out.print("<img align=\"center\" src=\"images/up.gif\">");
   else
        out.print("<img src=\"images/"+status.getCriticalityString()+".gif\">");
   out.print("<a href=\"device.jsp?id="+webmngr.getInfoDevice(names[i]).getName()+"\">");
   out.print("webmngr.getInfoDevice(names[i]).getFunctionalDescription() + "<BR>");
} %>
</body></html>
```

Figura 4.5 – Modelo de página JSP

Capítulo 5

Uso Prático da Solução: Gerenciando a rede do POP-Pb

Este capítulo demostra a aplicabilidade da solução desenvolvida através de um exemplo prático, onde a configuração realizada para este exemplo especifica uma visão para rede do POP-Pb (*Point of Presence – Paraiba*). Faremos então uma rápida excursão pela aplicação, acreditando que este é o modo menos tedioso de demostrar o seu funcionamento.

5.1 A rede do POP-Pb

O POP-PB localiza-se no primeiro andar do prédio do Departamento de Sistemas e Computação (bloco CN) na Universidade Federal da Paraíba – Campus II, e encontra-se atualmente sob a coordenação administrativa do professor Mário Assad, da Universidade Federal da Paraíba - Campus I e coordenação técnica do professor Pedro Sérgio Nicolletti, da Universidade Federal da Paraíba - Campus II.

O POP-PB tem como principal responsabilidade manter a Rede Paraibana de Pesquisa (RPP) em perfeito funcionamento. Dentre outras instituições, fazem parte da rede paraibana de pesquisa todos os campi da Universidade Federal da Paraíba (Patos, Areia, Cajazeiras, Campina Grande e João Pessoa), o Centro Federal de Educação Tecnológica da Paraíba (CEFET-PB) e a Fundação Parque Tecnológico.

Diante dessa responsabilidade, o POP-Pb precisa prover um serviço com qualidade para atender satisfatoriamente aos seus usuários. Alguns dos equipamentos envolvidos neste serviço e que precisam ser gerenciados estão relacionados na Tabela 5.1. Passaremos agora a descrever a solução, mostrando sua aplicabilidade para rede do POP-Pb.

Quantidade	Tipo de equipamento	Finalidade
01	Roteador Cisco Série 2500	Roteador federal (RNP) para a integração da Rede Paraibana de Pesquisa (RPP) à Rede Nacional de Pesquisa (RNP)/Internet
02	Roteador IBM 2210	Roteadores para distribuir acesso Internet no Estado da Paraíba (para instituições de ensino e pesquisa)
02	IBM RISC System/6000 4015	Servidores Proxy Web
02	IBM RISC Systems/6000 240	Controladores dos servidores Proxy Web
01	IBM RISCSystem/6000 43P-240	Servidor de nomes, correio eletrônico, FTP e Web do PoP-PB
01	Switch IBM 8271 - 712	Responsável pela integração dos equipamentos do POP-PB

Tabela 5.1 - Equipamentos do POP-PB

5.2 Gerenciando a Rede com a Aplicação

Após sua devida instalação e configuração, a solução de gerência pode mostrar a rede graficamente para o operador através de uma interface Web (usando qualquer um dos browsers disponíveis na Internet). A figura 5.1 mostra a página inicial da aplicação, onde pode ser observada uma lista contendo todos os nomes das redes gerenciadas. Para iniciar a navegação, é preciso clicar sobre o nome da rede, ou selecionar a opção no menu suspenso, encontrado no *frame* superior da página Web. Iniciaremos nossa navegação através do mapa principal do POP-Pb (Figura 5.2), que mostra quatro pontos de presença da Internet no Brasil: O POP-Pb, situado em Campina Grande, o POP-Pe situado em Recife, o POP-SP situado em São Paulo, e o POP- RJ situado no Rio de Janeiro.

Figura 5.1: Página inicial da aplicação

Como especificado na seção 3.5, a representação ou visão da rede é construída por um conjunto de mapas relacionados, onde cada mapa representa uma determinada parte do sistema. Um mapa pode mostrar vários objetos gráficos ou figuras para representar os elementos da rede, que por sua vez podem possuir *hiperlinks* para outros mapas (sub-mapas) ou páginas contendo informações detalhadas sobre os equipamentos gerenciados.

Em comparação com o protótipo da aplicação (seção 3.1), a nova versão traz poucas novidades quanto ao conteúdo das informações apresentadas na interface. Basicamente, a aplicação continua apresentando mapas da rede, informações sobre os equipamentos gerenciados e gráficos estatísticos sobre variáveis. A indicação do estados dos equipamentos continua sendo realizada pelo uso de cores: um ponto verde sobre um equipamento ou canal de comunicação significa um estado operacional correto; pontos pretos, cinzas, laranjas, amarelos e vermelhos indicam problemas de diferentes níveis críticos, que podem ser propagados entre os mapas da rede.

Uma novidade apresentada nesta versão é que não existe mais dependência hierárquica entre os mapas da rede, sendo sua estrutura melhor representada através de um

grafo[1]. Entretanto, a grande e principal diferença consiste na velocidade de atualização dessas informações, apresentadas através de páginas HTML geradas dinamicamente pela aplicação. Utilizando este recurso, é possível refletir o estado da rede na interface com um tempo máximo de latência determinado em segundos (uma boa configuração está na faixa de cinco segundos, mais ou menos).

Figura 5.2: Mapa principal da rede do POP-Pb

O mapa principal destaca problemas na rede do POP-Pb. Clicando sobre o ponto em vermelho, seremos levados para outro mapa mais detalhado (Figura 5.3) sobre esta parte da rede (imagine que os mapas possuem sensibilidade à posição do mouse). Neste mapa, são apresentados os seguintes elementos principais: o roteador do POP-Pb, o canal de comunicação do roteador do POP-Pb à RNP, e três sub-mapas representado a RNP (mapa anterior), a RPP (Rede Paraibana de Pesquisa) e o Serviço Web Proxy do POP-Pb.

[1] Um grafo consiste de um conjunto de nós e num conjunto de arcos, sendo cada arco do grafo representado por um par de nós. No nosso caso, cada nó representa um mapa distinto da rede, e cada arco especifica um par de mapas interligados através de um *hiperlink*.

Figura 5.3: Mapa detalhado do POP-Pb

A figura 5.3 indica problemas críticos no serviço Web Proxy do POP-Pb. Ao clicarmos sobre a figura do Switch POP-Pb, seremos levados para outro mapa que detalha esta parte da rede (figura 5.4). No mapa Serviço Web Proxy, aparece o estado operacional do Switch IBM 8271 e dos equipamentos que formam o sistema de cache[2] do POP-Pb.

Até aqui, já deve estar claro que a mudança de algum estado operacional dentro de um mapa pode ser propagada para seus sub-mapas, permitindo a identificação do problema a partir de outro ponto da rede. No caso da existência de mais de um problema com diferentes níveis críticos num mesmo mapa, podemos escolher qual o nível que será propagado na configuração da aplicação (seção 3.3.6). Neste exemplo, o problema foi identificado como sendo no Switch IBM 8271 do POP-Pb. Clicando sobre o Switch IBM

[2] Este sistema oferece uma maneira eficiente de melhorar o tempo de acesso as páginas e os arquivos mais requisitados pela Web, armazenando cópia dessas informações em disco. Dessa forma, quando esta informação for requisitada novamente, é fornecida a cópia já gravada localmente.

8271, um novo mapa será apresentado, mostrando o estado operacional do equipamento e também de suas interfaces. Este novo mapa indica que o problema esta localizado exatamente em uma das interfaces do equipamento (figura 5.5).

Figura 5.4: Mapa Serviço Web Proxy do POP-Pb

Clicando na interface, o operador pode obter mais informações sobre o elemento gerenciado e acessar gráficos estatísticos sobre seu comportamento durante toda a semana (figura 5.6). Preenchendo os formulários existentes na página dos elementos e submetendo os pedidos ao servidor Web da aplicação, o operador pode obter gráficos em outra data específica sobre uma determinada estatística disponível (figura 5.7).

Figura 5.5: Mapa mostrando o Switch IBM 8271 do POP-PB

Além dos mapas e páginas mostrando informações sobre os elementos gerenciados, o operador pode ter acesso diretamente à relação dos equipamentos, que mostra seus respectivos estados de funcionamento (figura 5.8). A partir dessa lista, pode-se acessar um relação dos alarmes ocorridos na rede relacionados com cada elemento (Figura 5.9).

Figura 5.6: Informações sobre a interface do Switch IBM 8271 do POP-PB

Figura 5.7: Gráfico sobre o tempo de ping do Switch 8271

Além da indicação dos alarmes nas páginas Web, a aplicação também pode executar envio de *e-mails* e chamadas telefônicas para o celular do operador da rede, indicando quando algum problema é detectado durante o processo de monitoração. Este é um recurso sofisticado, sendo também bastante importante para ajudar a corrigir os problemas da rede antes que os usuários venham a sentir as conseqüências.

Figura 5.8: Lista de equipamentos gerenciados pela aplicação

Figura 5.9 : Lista de alarmes do equipamento do Switch IMB 8271

Capítulo 6

Conclusões

Neste trabalho, especificamos e implementamos uma solução cujo objetivo principal é facilitar as atividades dos gerentes e administradores de redes de computadores. Trata-se de uma aplicação de gerência baseada em Web, independente de plataforma, utilizando uma arquitetura em três camadas e componentes reutilizáveis de software para atender de modo adequado as necessidades desses usuários. Neste capítulo, apresentamos os resultados obtidos, mostrando como foram atendidas as insatisfações originais. Também demonstramos como podemos alcançar os requisitos que não foram completa ou parcialmente satisfeitos e apresentamos os trabalhos futuros.

6.1 Avaliação e Satisfação dos Requisitos

Vimos que as soluções de gerência tradicionais apresentam problemas que impõem sérias limitações ao desenvolvimento das atividades de gerência. Assim, ao levantar os requisitos que caracterizam uma solução mais adequada, fomos capazes de especificar o projeto de uma aplicação de gerência que oferece um conjunto de mecanismos sofisticados para atender os objetivos propostos.

Diante dos requisitos levantados, propomos uma arquitetura que fornece além de reutilização de código, independência de plataforma e integração com Web, uma grande capacidade de modularização e flexibilidade para adição de novas funcionalidades.

Finalizada a fase de implementação e testes desta arquitetura, passamos agora a descrever como foram atendidos os requisitos levantados. A tabela 6.1 relaciona os requisitos atendidos com as características apresentadas pela solução:

Requisito	Característica da Solução
F1	Componentes do módulo `Monitor`.
F3	Componentes do módulo `Log`.
F5	Componentes dos módulos `EventGenerator e Status`.
F6	Componentes do módulo `Log`.
F8	Componentes do módulo `Mailer`.
F12	Conjunto de mapas relacionados por *hiperlinks*, onde cada mapa representa uma determinada parte da rede.
F13	Indicação do estado da rede pelo uso de cores associadas à criticalidade dos equipamentos configurados.
F14	Componentes do módulo `Servlet`, instanciados pelo servlet JSP para acessar as informações apresentadas nas páginas Web.
F15	Página Web contendo uma lista dos equipamentos gerenciados pela aplicação, e seus respectivos estados de funcionamento.
F17	Componentes gráficos do módulo `Servlet`, instanciados pelo servlet JSP para acessar as informações apresentadas nas páginas Web.
F23	Inserção de novos arquivos MIB diretamente no diretório apropriado.
F24 – F27	Componentes do módulo `Config`, instanciados de acordo com as definições estabelecidas nos arquivos XML de configuração.
NF1 – NF3	API Snmpv3 disponibilizada pela AdventNet [AdventNet, 2000].
NF4 – NF5	Arquitetura seguindo um modelo em três camadas.
NF6 – NF8	Utilização da tecnologia Java da Sun Microsystems [Sun Microsystems, 2000] para permitir portabilidade e integração entre as diferentes camadas da aplicação.
NF9 – N11	Páginas JSP geradas dinamicamente pela aplicação.
NF12 – F13	Publicação HTML distribuída junto com a aplicação contendo manual do usuário, guia de instalação e API Javadoc.
NF14	Empacotamento da aplicação utilizando um arquivo ZIP.

Tabela 6.1 - Requisitos atendidos pela solução

Como indicado na tabela anterior, a solução desenvolvida atendeu grande parte dos requisitos levantados para o projeto. Basicamente, a aplicação conseguiu oferecer todas as funcionalidades encontradas no protótipo analisado, porém adicionando importantes melhorias devido à reformulação da arquitetura utilizada pela solução. Em relação à arquitetura proposta, podemos destacar os seguintes pontos bem sucedidos:

Desenvolvimento em JAVA

Um ambiente de rede envolve diferentes tipos de equipamentos, com diferentes sistemas operacionais. Assim, as tecnologias utilizadas devem ser bastantes flexíveis, facilitando a portabilidade da aplicação para diferentes plataformas. Para atender este requisito, a nova versão foi desenvolvida utilizando o padrão Java (TM) oferecido pela Sun Microsystems. A portabilidade alcançada pelo uso dessa tecnologia facilita bastante a migração da aplicação para diferentes plataformas, permitindo ainda amplo suporte para programação em rede e integração com a Web.

Arquitetura Baseada em Componentes (Java Beans)

A nova versão foi inteiramente construída utilizando JavaBeans, que são componentes reutilizáveis de software cujos métodos e propriedades podem ser reconhecidas em tempo de execução, e manipuladas dinamicamente dentro de aplicações. Esses componentes tanto podem ser encaixados para adição de novas funcionalidades ou utilizados para criação de outros JavaBeans. Utilizando essa abordagem foi possível construir uma aplicação modularizada e extensível, facilitando assim o cumprimento dos requisitos levantados.

Configuração da Aplicação

A configuração da nova versão foi modularizada através de componentes que encapsulam em suas propriedades as informações necessárias ao funcionamento da aplicação. A solução utiliza XML para especificar esses componentes, permitindo ao usuário realizar as configurações necessárias através de editores gráficos XML disponíveis no mercado, a exemplo do Xeena da IBM. Tal possibilidade permite uma melhor visualização dos componentes da rede e facilita o processo de edição de suas propriedades.

Suporte para MIB, versão 1 e 2

Agentes SNMP provêem acesso para variáveis específicas ao tipo de dispositivo gerenciado. Dessa forma, é necessário conhecer as variáveis que são suportadas pelos equipamentos. A nova versão da aplicação utiliza um compilador para carregar

automaticamente os módulos MIB e interpretar as informações obtidas nos agentes, diminuindo a quantidade de informações necessárias para sua configuração.

Monitoração dos Dispositivos via Threads

Threads ou linhas de execução é uma maneira de implementar paralelismo dentro de processos. Como forma de melhorar seu desempenho, resolvendo um dos maiores problemas do protótipo da solução, a solução utiliza este recurso para gerenciar os equipamentos, criando threads independentes para monitorar as variáveis de cada dispositivo na rede.

Modelos de Páginas JSP

O uso de modelos JSP com código Java embutido facilitou o processo de desenvolvimento e configuração da nova versão. Utilizando esta arquitetura de aplicação, conseguimos isolar a funcionalidade que queremos dar ao sistema (*Business Logic*) da sua interface (*Presentation Logic*), facilitando também sua manutenção futura.

Estrutura dos Mapas

A disposição dos mapas da rede através de uma estrutura em grafo, fornecida pela nova versão da aplicação, foi mais apropriada do que a estrutura hierárquica imposta pelo protótipo. Isso permite a presença de um mesmo elemento em diferentes mapas da rede.

Desempenho e Atualidade das Informações

Com respeito à velocidade, usar um browser para acessar páginas Web geradas dinamicamente (mapas, imagens e gráficos) permite uma visualização rápida e atualizada sobre o estado da rede, sem sobrecarregar a estação de gerência durante a geração dessas informações. Outra vantagem é que esta interface permite acesso remoto de qualquer ponto da rede, mesmo com a presença de um *firewall*.

6.2 Problemas Enfrentados

É muito importante destacarmos as dificuldades enfrentadas neste trabalho para repassarmos toda experiência obtida com sua realização. A maior dificuldade enfrentada foi a nossa falta de experiência com análise e projeto de sistemas orientado à objetos, principalmente quanto aos aspectos gerenciais do desenvolvimento de software.

Durante o projeto, contamos com a ajuda de vários alunos do Departamento de Sistemas e Computação (DSC) da UFPB, e constatamos que esse tipo de dificuldade também se estendia à maioria das pessoas envolvidas, o que atrapalhou bastante o ritmo

dos trabalhos. Além disso, iniciamos o projeto aplicando uma metodologia que não era apropriada, baseada num processo de desenvolvimento seguindo o modelo em cascata. Trabalhando dessa forma, fomos obrigados a levantar todos riscos e problemas que poderiam surgir durante o projeto, mesmo sem ter experiência suficiente para isso.

Na metade do tempo estipulado ao projeto, sentimos grande necessidade de mudar nossa forma de trabalho. Procuramos então utilizar outro método que oferecesse resultados rápidos, porém substanciais. Tentar gerar o sistema por inteiro, no final do processo, parecia agora uma maneira errada e perigosa de progredir. Desse modo, um processo iterativo e incremental se tornou a maneira mais natural de continuar o trabalho.

Esta mudança permitiu delinear uma arquitetura base flexível, cujas funcionalidades poderiam ser incrementadas através de iterações. No final de cada iteração, era gerado um software de qualidade, que atendia um conjunto maior de requisitos dentre aqueles levantados inicialmente. Porém, antes de ser implementada, cada nova funcionalidade era devidamente analisada e incorporada da melhor forma possível, sendo o projeto final formado pelo conjunto de todos os incrementos realizados.

Utilizando esta metodologia, conseguimos atender boa parte dos objetivos planejados, mas infelizmente não conseguimos atender a todos eles por um questão de tempo. Porém, na busca por melhores soluções, acreditamos que várias lições foram assimiladas, e esperamos que todas as pessoas envolvidas no projeto tenham obtido uma melhor visão das técnicas adequadas e dos aspectos gerenciais envolvidos no desenvolvimento de software.

6.3 Trabalhos Futuros

Destacamos agora os requisitos que não foram abordados na solução atual, mas que podem ser adicionados através de incrementos na arquitetura proposta. Certamente, estes serão os principais pontos abordados para a produção de novas versões da aplicação:

F2 – Notificações: não foi oferecido nenhum suporte à notificação automática de eventos pelos agentes espalhados na rede. Um servidor de *traps* SNMP deverá ser usado para diminuir o tempo de identificação de problemas e minimizar o tráfego gerado pelo protocolo SNMP na rede;

F4 - Recuperação da Informação: os gráficos estatísticos só geram curvas diretamente dos valores obtidos das variáveis MIB. Um mecanismo para permitir cálculos gerais envolvendo essas variáveis deverá ser implementado;

F7 - Correlação de Alarmes: a aplicação não realiza a correlação de alarmes. Como exemplo, se um roteador sofrer uma pane, vários alarmes iriam ser disparados ao mesmo tempo (roteador e interfaces), sem a indicação de uma causa comum. Um mecanismo deverá ser implementado para oferecer suporte à notificação automática de eventos pelos agentes espalhados na rede.

F9, F16 – Relatórios: deveria haver facilidades para geração de relatórios, sendo possível visualizar estatísticas selecionadas de diferentes equipamentos e incluindo a geração e o envio automático via e-mail;

F10 - Parametrização dos Equipamentos: a aplicação deverá dar suporte à configuração remota dos equipamentos, utilizando programas ou métodos mais seguros que a solução provida pelo SNMP (primitiva *set*);

F11 - Autotopologia: deverá existir um mecanismo para reconhecimento automático da configuração da rede, reduzindo a necessidade de configuração manual. Três aspectos deverão ser considerados: descobrimento de equipamentos presentes na rede, identificação dos canais de comunicação entre estes equipamentos e geração automática dos mapas para visualização da rede;

F18 - MIB *Browser*: uma aplicação simples deverá ser provida para visualizar qualquer variável MIB de um determinado equipamento;

F19, F20 e F21 - Acesso à Aplicação: nenhuma tipo de controle de acesso a aplicação foi implementado nesta versão. Um mecanismo para atribuição de senhas e identificação de usuários poderá ser implementado;

F22 a F28 - Interatividade e Configuração: a interface provê pouca interatividade, principalmente com respeito à escolha de variáveis, escala de tempo e estilo dos gráficos. Além disso, mecanismo visuais (se possível automatizados), para todos os aspectos de configuração poderão ser providos junto com a solução.

Através da adição de novos elementos dentro da arquitetura desenvolvida os requisitos acima poderiam ser facilmente incorporados na aplicação. Várias extensões poderiam ser realizadas (como por exemplo, componentes RMON) para aprimorar esta arquitetura, na busca de soluções adequadas para o desenvolvimento de uma plataforma de

gerência baseada em Web. Com o tempo, poderíamos prover uma estrutura adequada ao desenvolvimento e execução de aplicações, um *framework* [Johnson & Foote, 1991] de gerência de redes. Atualmente, a aplicação se apresenta suficiente para monitoração de redes de pequeno a médio porte, com dezenas de equipamentos gerenciados.

Para demonstrar a flexibilidade da arquitetura proposta, tomamos como exemplo o requisito **F2.** Na Figura 6.1, especificamos em alto-nível como este requisito poderia ser incrementado à solução. Para oferecer suporte a notificação, nossa arquitetura precisaria incorporar um componente Servidor de *Traps* SMNP, que seria responsável por receber notificações sobre eventos interessantes pelos agentes espalhados na rede. Esse componente estaria agregado ao Infobus de Eventos, produzindo informações correspondentes aos eventos notificados. Desse modo, as devidas providências para alertar o usuário sobre sua ocorrência poderiam ser tomadas pela aplicação de gerência.

Figura 6.1 – Arquitetura com Servidor de Traps SNMP

Bibliografia

[AdventNet, 1999] AdventNet Inc., *On-line*: http://www.adventnet.com/

[Almeida e Souza, 2000a] Almeida, R. R., Souza, A. D. D., *API JConfig: Pacote de Configuração de JavaBeans em XML*, 2000. *On-line*: http://vulcano.dsc.ufpb.br/rodrigor/

[Almeida e Souza, 2000b] Almeida, R. R., Souza, A. D. D., *Um Sistema de Navegação em Redes Baseado em Web*, 2000. *On-line*: http://vulcano.dsc.ufpb.br/rodrigor/

[Andrade e Sauvé, 1998] Andrade, A. F. B., Sauvé, J. P., "*Directions in Network Management: Web-Based Solutions"*, 1998. *On-line*: http://vulcano.dsc.ufpb.br/hp-grc/producao/rt/rt-grc-1998-5/rt-grc-1998-5.html

[Armstrong, 1999] Armstrong. E., *Working with XML*, 1999. *On-line*: http://industry.ebi.ac.uk/~senger/xml/docs/tutorial/

[Case *at al*., 1990] Case, J. D., Fedor, M. S., Schoffstall, M. L., Davin, J. R., *A Simple Network Management Protocol*. Request for Comments 1157, DDN Network Information Center, SRI International, May 1990.

[Case *at al*., 1993] Case, J. D., MacCloghrie, K., Rose, M. T., Waldbusseer, S., *Introduction to version 2 of the Internet-standard Network Management Framework.* Request for Comments 1441, SNMP Research Inc., Hughes LAN Systems Inc., Dover Beach Consulting Inc., Carnegie Mellon University, April 1993.

[Case *at al*., 1998] Case, J. D., Mundy, R., Partain, D., Stewart, B., *Introduction to version 3 of the Internet-standard Network Management Framework*, Internet-draft, 1998.

[Colan, 1998] Colan, M., Karle, J., . T. Corcoran, *Let InfoBus Plug Your Beans Together*, 1998, On-line: http://www.devx.com/upload/free/features/javapro/1998/02feb98/mc0298/mc0298.asp

[Corcoran, 1996] C. T. Corcoran, *Managing Network Ills Network Managers Tackle Management-Tools Deficiencies With Web Technologies*, *InfoWorld*, 18(42), 1996.

[DMTF, 1999] DMTF – Distributed Management Task Force, *DMTF Desktop Management Interface Specification.* 1999, *On-line*: http://www.dmtf.org/wbem/

[Ghetie, 1997] I. G. Ghetie, *Network and Systems Management: Platform Analysis and Evaluation*, Kluwer Academic Publishing, 1997.

[Hyde, 1996] Hyde, D., *"The New Paradigm for Network Management"*, 1996. *On-line:* http://www.3com.com/technology/tech_net/white_papers/500627.html.

[Huntington-lee, 1997] Huntington-Lee, Case, J. D., *HP's OpenView: A Manager's Guide*, McGraw Hill, 1997.

[InfoBus, 1999] Sun Microsystems Inc., *On-line*: http://www.javasoft.com/beans/infobus

[ISO/IEC 7498:1984] *Information Processing Systems - Open Systems Interconnection: Basic Reference Model.* International Organization for Standardization and International Electrotechnical Committee, International Standard 7498, 1984.

[ISO/IEC 8824:1987] *Information Processing Systems - Open Systems Interconnection: Specification of Abstract Syntax Notation One (ASN.1).* International Organization for Standardization and International Electrotechnical Committee, International Standard 8824, December 1987.

[JConfig, 2000] Almeida, R. R., Souza, A. D. D., *API JConfig: Pacote de Configuração de JavaBeans em XML*, 2000. *On-line*: http://vulcano.dsc.ufpb.br/rodrigor/

[Johnson & Foote, 1991] Johnson, R. E., Foote, B., *Designing Reusable Classes*. Journal of Object-Oriented Programming, June/July 1991.

[Gamma e Beck, 1999] Gamma, E., Beck, K., *Test Infected: Programmers Love Writing Tests*, 1999. *On-line*: http://members.pingnet.ch/gamma/junit.htm

[Larsen, 1996] A. K. LARSEN, *"The Next Web Wave: Network Management"*, *Data Communications*, 25(01), 1996.

[Lopes, 2000] Lopes, R. V., *Uma API Java para Geração de Gráficos Estatíticos*, 2000. *On-line*: http://www.dsc.ufpb.br/~rakel/

[Meira e Lages, 1998] Meira, D. M., Larges, N. A. C., *SIS: Um Sistema Integrado de Supervisão para Telecomunicações.* Relatório Técnico, TELEMIG/UFMG/88, Depto. De Informática, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, Setembro 1988.

[Rofail & Shohoud, 1999] Rofail, A., Shohoud, Y., *Mastering COM and COM +*. Fourth Edition, Sybex, 1999.

[Rogers, 1996] J. A. Rogers, "Browser Power", 1996. *On-line*: http://www.techweb.com/se/directlink.cgi?INW19971117S0071P.

[Rose, 1990] Rose, M. T., *The Simple Book: An Introduction To Networking Management of TCP/IP-based Networks*, Prentice Hall, 1990.

[Rose & Mccloghrie, 1990a] Rose, M. T., McCloghrie, K., *Management Information Base Network Management of TCP/IP based Internets.* Request for Comments 1156, DNN Network Information Center, SRI International, May 1990.

[Rose & Mccloghrie, 1990b] Rose, M. T., McCloghrie, K., *Structure and Identification of Management Information for TCP/IP based Internets.* Request for Comments 1155, DNN Network Information Center, SRI International, May 1990.

[Rose & Mccloghrie, 1995] Rose, M. T., McCloghrie, K., *How to Manage Your Network Using SNMP: The Networking Management Practicum*, Prentice Hall, 1995.

[Rose, 1996] Rose, M. T., *Industry Comment.* The Simple Times, vol.4, nº3, pp 22-31, 1996.

[Rumbaugh *et al*., 1999] Rumbaugh, J., Booch, G., Jacobson, I., *The Unifield Software Development Process*. Addison-Wesley, 1999.

[Sloman, 1994] Sloman, M., *Network and Distributed Systems Management*. Addison-Wesley, 1994.

[Sun Microsystems, 1999] Sun Microsystems Inc. *Java Management Extensions – SNMP Manager API*. August, 1999. Draft 2.0.

[Sun Microsystems, 2000] Sun Microsystems Inc., *On-line*: http://java.sun.com/

[Smith, 1996] M. Smith, *"Enterprise Management Glue"*, 1996. *On-line:* http://kinetworks.ki.com/WBEM/emglue.html.

[Sauvé, 1998] Sauvé, J. P. Sauvé, *WebManager: Uma Plataforma de Gerência de Redes Baseada em Web*, 1998. *On-line:* http://vulcano.dsc.ufpb.br/hp-grc/producao/rt/rt-grc-1998-2/rt-grc-1998-2.html

[Sauvé, 1999] Sauvé, J. P., *WebManager: A Web-Based Network Management Application.* LANOMS – Latin American Network Operations and Management Symposium, Rio de Janeiro, Brasil, Dezembro 1999.

[Siegel, 1996] Siegel, J., *Corba Fundamentals and Programming*, John Wiley & Sons, 1996.

[Sommerville, 1992] Sommerville, I., *Software Engineering*. Addison-Wesley, 1992.

Apêndice A

Configuração XML para os Equipamentos do POP-Pb

```
<?xml version="1.0" encoding="UTF-8"?>
<webMngrConfig>
  <NetEquipment>
    <name>bb2</name>
    <hardwareDescription>Cisco 2501</hardwareDescription>
    <functionalDescription>Roteador Principal do POP-PB</functionalDescription>
    <criticality>9</criticality>
    <host>bb2.pop-pb.rnp.br</host>
    <port>161</port>
    <protocol>0</protocol>
    <retries>0</retries>
    <timeout>4000</timeout>
    <pollInterval>5</pollInterval>
    <pollsToSave>3</pollsToSave>
    <community>RNP</community>
    <writeCommunity>private</writeCommunity>
    <instanceIndex>0</instanceIndex>
    <statusOID>.webmngr.ping.PingUp</statusOID>
    <allGraphics>
      <variableGraphic>
        <title>Status Operacional</title>
        <oid>.webmngr.ping.PingUp</oid>
        <label>Fracao do tempo Up</label>
        <limit>1</limit>
      </variableGraphic>
      <variableGraphic>
        <title>Tempo de Ping</title>
        <oid>.webmngr.ping.PingTime</oid>
        <label>ms</label>
        <limit>1000</limit>
      </variableGraphic>
```

```
      <variableGraphic>
        <title>Memoria Livre</title>
        <oid>.iso.org.dod.internet.private.enterprises.cisco.local.lsystem.freeMem</oid>
        <label>bytes</label>
        <limit>16777216</limit>
      </variableGraphic>
      <variableGraphic>
        <title>Utilizacao recente da CPU</title>
        <oid>.iso.org.dod.internet.private.enterprises.cisco.local.lsystem.avgBusy5</oid>
        <label>Percentual Ocupado</label>
        <limit>100</limit>
      </variableGraphic>
      <variableGraphic>
        <title>Buffers nao criados por falta de memoria</title>
        <oid>.iso.org.dod.internet.private.enterprises.cisco.local.lsystem.bufferNoMem</oid>
        <label>Numero de Buffers</label>
        <limit>20</limit>
      </variableGraphic>
    </allGraphics>
    <Interfaces>
      <interface>
        <name>bb2_s0</name>
        <index>2</index>
        <bandwidth>1024000</bandwidth>
        <hardwareDescription>Serial 0 / Cisco 2501</hardwareDescription>
        <functionalDescription>Enlace do POP-PB/CPV ao POP-PE/REC (1
Mbps)</functionalDescription>
        <statusOID>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</statusOID>
        <allGraphics>
          <variableGraphic>
            <title>Status Operacional</title>
            <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</oid>
            <label>Fracao do tempo Up</label>
            <limit>1</limit>
          </variableGraphic>
          <variableGraphic>
            <title>Numero de Resets</title>
            <oid>.iso.org.dod.internet.private.enterprises.cisco.local.linterfaces.lifTable.
lifEntry.locIfResets</oid>
            <label>Resets</label>
            <limit>20</limit>
          </variableGraphic>
          <variableGraphic>
            <title>Taxa de Entrada</title>
            <oid>.iso.org.dod.internet.private.enterprises.cisco.local.linterfaces.lifTable.
lifEntry.locIfInBitsSec</oid>
            <label>Numero de Buffers</label>
            <limit>1024000</limit>
```

```
        </variableGraphic>
        <variableGraphic>
          <title>Taxa de Saida</title>
          <oid>.iso.org.dod.internet.private.enterprises.cisco.local.linterfaces.lifTable.
lifEntry.locIfOutBitsSec</oid>
          <label>Numero de Buffers</label>
          <limit>1024000</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Numero de Colisoes</title>
          <oid>.iso.org.dod.internet.private.enterprises.cisco.local.linterfaces.lifTable.
lifEntry.locIfCollisions</oid>
          <label>Colisoes</label>
          <limit>100</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Recebidos</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInErrors</oid>
          <label>pacotes</label>
          <limit>100</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Enviados</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutErrors</oid>
          <label>pacotes</label>
          <limit>100</limit>
        </variableGraphic>
      </allGraphics>
    </interface>
  </Interfaces>
 </NetEquipment>
 <NetEquipment>
  <name>atmSP</name>
  <hardwareDescription>IBM 826x</hardwareDescription>
  <functionalDescription>Comutador ATM do POP-SP (Sao Paulo/SP)</functionalDescription>
  <criticality>9</criticality>
  <host>sp.bb3.rnp.br</host>
  <port>161</port>
  <protocol>0</protocol>
  <retries>0</retries>
  <timeout>4000</timeout>
  <pollInterval>5</pollInterval>
  <pollsToSave>3</pollsToSave>
  <community>RNP</community>
  <writeCommunity>private</writeCommunity>
  <instanceIndex>0</instanceIndex>
  <statusOID>.webmngr.ping.PingUp</statusOID>
  <allGraphics>
```

```
      <variableGraphic>
        <title>Status Operacional</title>
        <oid>.webmngr.ping.PingUp</oid>
        <label>Fracao do tempo Up</label>
        <limit>1</limit>
      </variableGraphic>
      <variableGraphic>
        <title>Tempo de Ping</title>
        <oid>.webmngr.ping.PingTime</oid>
        <label>ms</label>
        <limit>1000</limit>
      </variableGraphic>
    </allGraphics>
  </NetEquipment>
  <NetEquipment>
    <name>atmRJ</name>
    <hardwareDescription>IBM 826x</hardwareDescription>
    <functionalDescription>Comutador ATM do POP-RJ (Rio de
Janeiro/RJ)</functionalDescription>
    <criticality>9</criticality>
    <host>rj.bb3.rnp.br</host>
    <port>161</port>
    <protocol>0</protocol>
    <retries>0</retries>
    <timeout>4000</timeout>
    <pollInterval>5</pollInterval>
    <pollsToSave>3</pollsToSave>
    <community>RNP</community>
    <writeCommunity>private</writeCommunity>
    <instanceIndex>0</instanceIndex>
    <statusOID>.webmngr.ping.PingUp</statusOID>
    <allGraphics>
      <variableGraphic>
        <title>Status Operacional</title>
        <oid>.webmngr.ping.PingUp</oid>
        <label>Fracao do tempo Up</label>
        <limit>1</limit>
      </variableGraphic>
      <variableGraphic>
        <title>Tempo de Ping</title>
        <oid>.webmngr.ping.PingTime</oid>
        <label>ms</label>
        <limit>1000</limit>
      </variableGraphic>
    </allGraphics>
  </NetEquipment>
  <NetEquipment>
    <name>atmPE</name>
    <hardwareDescription>IBM 826x</hardwareDescription>
    <functionalDescription>Comutador ATM do POP-PE (Recife/PE)</functionalDescription>
```

```
  <criticality>9</criticality>
  <host>pe.bb3.rnp.br</host>
  <port>161</port>
  <protocol>0</protocol>
  <retries>0</retries>
  <timeout>4000</timeout>
  <pollInterval>5</pollInterval>
  <pollsToSave>3</pollsToSave>
  <community>RNP</community>
  <writeCommunity>private</writeCommunity>
  <instanceIndex>0</instanceIndex>
  <statusOID>.webmngr.ping.PingUp</statusOID>
  <allGraphics>
    <variableGraphic>
      <title>Status Operacional</title>
      <oid>.webmngr.ping.PingUp</oid>
      <label>Fracao do tempo Up</label>
      <limit>1</limit>
    </variableGraphic>
    <variableGraphic>
      <title>Tempo de Ping</title>
      <oid>.webmngr.ping.PingTime</oid>
      <label>ms</label>
      <limit>1000</limit>
    </variableGraphic>
  </allGraphics>
</NetEquipment>
<NetEquipment>
  <name>ciscoPE</name>
  <hardwareDescription>Cisco 7000</hardwareDescription>
  <functionalDescription>Roteador do POP-PE (Recife/PE)</functionalDescription>
  <criticality>9</criticality>
  <host>bb2.pop-pe.rnp.br</host>
  <port>161</port>
  <protocol>0</protocol>
  <retries>0</retries>
  <timeout>4000</timeout>
  <pollInterval>5</pollInterval>
  <pollsToSave>3</pollsToSave>
  <community>RNP</community>
  <writeCommunity>private</writeCommunity>
  <instanceIndex>0</instanceIndex>
  <statusOID>.webmngr.ping.PingUp</statusOID>
  <allGraphics>
    <variableGraphic>
      <title>Status Operacional</title>
      <oid>.webmngr.ping.PingUp</oid>
      <label>Fracao do tempo Up</label>
      <limit>1</limit>
    </variableGraphic>
```

```
      <variableGraphic>
        <title>Tempo de Ping</title>
        <oid>.webmngr.ping.PingTime</oid>
        <label>ms</label>
        <limit>1000</limit>
      </variableGraphic>
    </allGraphics>
  </NetEquipment>
  <NetEquipment>
    <name>sw8271</name>
    <hardwareDescription>IBM 8271</hardwareDescription>
    <functionalDescription>Switch POP-PB.RNP.BR</functionalDescription>
    <criticality>9</criticality>
    <host>200.129.64.132</host>
    <port>161</port>
    <protocol>0</protocol>
    <retries>0</retries>
    <timeout>4000</timeout>
    <pollInterval>5</pollInterval>
    <pollsToSave>3</pollsToSave>
    <community>RNP</community>
    <writeCommunity>private</writeCommunity>
    <instanceIndex>0</instanceIndex>
    <statusOID>.webmngr.ping.PingUp</statusOID>
    <allGraphics>
      <variableGraphic>
        <title>Status Operacional</title>
        <oid>.webmngr.ping.PingUp</oid>
        <label>Fracao do tempo Up</label>
        <limit>1</limit>
      </variableGraphic>
      <variableGraphic>
        <title>Tempo de Ping</title>
        <oid>.webmngr.ping.PingTime</oid>
        <label>ms</label>
        <limit>1000</limit>
      </variableGraphic>
    </allGraphics>
    <Interfaces>
      <interface>
        <name>sw8271_e3</name>
        <index>3</index>
        <bandwidth>1000000</bandwidth>
        <hardwareDescription>Ethernet 3 / IBM 8271</hardwareDescription>
        <functionalDescription>Enlace do POP-PB a UFPB/Cgrande</functionalDescription>
        <statusOID>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</statusOID>
        <allGraphics>
          <variableGraphic>
            <title>Status Operacional</title>
```

```
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</oid>
          <label>Fracao do tempo Up</label>
          <limit>1</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Numero Total de Octets Recebidos</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInOctets</oid>
          <label>Pacotes</label>
          <limit>64000</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Numero Total de Octets Enviados</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutOctets</oid>
          <label>Pacotes</label>
          <limit>64000</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Recebidos</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInErrors</oid>
          <label>Pacotes</label>
          <limit>100</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Enviados</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutErrors</oid>
          <label>Pacotes</label>
          <limit>100</limit>
        </variableGraphic>
      </allGraphics>
    </interface>
  </Interfaces>
</NetEquipment>
<NetEquipment>
  <name>nd1</name>
  <hardwareDescription>IBM RS6000/43P-240</hardwareDescription>
  <functionalDescription>Network Dispatcher 1</functionalDescription>
  <criticality>5</criticality>
  <host>nd1.pop-pb.cache.rnp.br</host>
  <port>161</port>
  <protocol>0</protocol>
  <retries>0</retries>
  <timeout>4000</timeout>
  <pollInterval>5</pollInterval>
  <pollsToSave>3</pollsToSave>
  <community>RNP</community>
```

```
  <writeCommunity>private</writeCommunity>
  <instanceIndex>0</instanceIndex>
  <statusOID>.webmngr.ping.PingUp</statusOID>
  <allGraphics>
    <variableGraphic>
      <title>Status Operacional</title>
      <oid>.webmngr.ping.PingUp</oid>
      <label>Fracao do tempo Up</label>
      <limit>1</limit>
    </variableGraphic>
    <variableGraphic>
      <title>Tempo de Ping</title>
      <oid>.webmngr.ping.PingTime</oid>
      <label>ms</label>
      <limit>1000</limit>
    </variableGraphic>
  </allGraphics>
</NetEquipment>
<NetEquipment>
  <name>nd2</name>
  <hardwareDescription>IBM RS6000/43P-240</hardwareDescription>
  <functionalDescription>Network Dispatcher 2</functionalDescription>
  <criticality>5</criticality>
  <host>nd2.pop-pb.cache.rnp.br</host>
  <port>161</port>
  <protocol>0</protocol>
  <retries>0</retries>
  <timeout>4000</timeout>
  <pollInterval>5</pollInterval>
  <pollsToSave>3</pollsToSave>
  <community>RNP</community>
  <writeCommunity>private</writeCommunity>
  <instanceIndex>0</instanceIndex>
  <statusOID>.webmngr.ping.PingUp</statusOID>
  <allGraphics>
    <variableGraphic>
      <title>Status Operacional</title>
      <oid>.webmngr.ping.PingUp</oid>
      <label>Fracao do tempo Up</label>
      <limit>1</limit>
    </variableGraphic>
    <variableGraphic>
      <title>Tempo de Ping</title>
      <oid>.webmngr.ping.PingTime</oid>
      <label>ms</label>
      <limit>1000</limit>
    </variableGraphic>
  </allGraphics>
</NetEquipment>
<NetEquipment>
```

```
  <name>proxy01</name>
  <hardwareDescription>IBM 6000/7015-R50</hardwareDescription>
  <functionalDescription>Proxy Server 1</functionalDescription>
  <criticality>7</criticality>
  <host>proxy01.pop-pb.cache.rnp.br</host>
  <port>161</port>
  <protocol>0</protocol>
  <retries>0</retries>
  <timeout>4000</timeout>
  <pollInterval>5</pollInterval>
  <pollsToSave>3</pollsToSave>
  <community>RNP</community>
  <writeCommunity>private</writeCommunity>
  <instanceIndex>0</instanceIndex>
  <statusOID>.webmngr.ping.PingUp</statusOID>
  <allGraphics>
    <variableGraphic>
      <title>Status Operacional</title>
      <oid>.webmngr.ping.PingUp</oid>
      <label>Fracao do tempo Up</label>
      <limit>1</limit>
    </variableGraphic>
    <variableGraphic>
      <title>Tempo de Ping</title>
      <oid>.webmngr.ping.PingTime</oid>
      <label>ms</label>
      <limit>1000</limit>
    </variableGraphic>
  </allGraphics>
</NetEquipment>
<NetEquipment>
  <name>proxy02</name>
  <hardwareDescription>IBM 6000/7015-R50</hardwareDescription>
  <functionalDescription>Proxy Server 2</functionalDescription>
  <criticality>7</criticality>
  <host>proxy02.pop-pb.cache.rnp.br</host>
  <port>161</port>
  <protocol>0</protocol>
  <retries>0</retries>
  <timeout>4000</timeout>
  <pollInterval>5</pollInterval>
  <pollsToSave>3</pollsToSave>
  <community>RNP</community>
  <writeCommunity>private</writeCommunity>
  <instanceIndex>0</instanceIndex>
  <statusOID>.webmngr.ping.PingUp</statusOID>
  <allGraphics>
    <variableGraphic>
      <title>Status Operacional</title>
      <oid>.webmngr.ping.PingUp</oid>
```

```
      <label>Fracao do tempo Up</label>
      <limit>1</limit>
    </variableGraphic>
    <variableGraphic>
      <title>Tempo de Ping</title>
      <oid>.webmngr.ping.PingTime</oid>
      <label>ms</label>
      <limit>1000</limit>
    </variableGraphic>
  </allGraphics>
</NetEquipment>
<NetEquipment>
  <name>rt2210</name>
  <hardwareDescription>IBM 2210</hardwareDescription>
  <functionalDescription>Roteador Distribuicao / POP-PB</functionalDescription>
  <criticality>9</criticality>
  <host>200.129.64.131</host>
  <port>161</port>
  <protocol>0</protocol>
  <retries>0</retries>
  <timeout>4000</timeout>
  <pollInterval>5</pollInterval>
  <pollsToSave>3</pollsToSave>
  <community>RNP</community>
  <writeCommunity>private</writeCommunity>
  <instanceIndex>0</instanceIndex>
  <statusOID>.webmngr.ping.PingUp</statusOID>
  <allGraphics>
    <variableGraphic>
      <title>Status Operacional</title>
      <oid>.webmngr.ping.PingUp</oid>
      <label>Fracao do tempo Up</label>
      <limit>1</limit>
    </variableGraphic>
    <variableGraphic>
      <title>Tempo de Ping</title>
      <oid>.webmngr.ping.PingTime</oid>
      <label>ms</label>
      <limit>1000</limit>
    </variableGraphic>
  </allGraphics>
  <Interfaces>
    <interface>
      <name>rt2210_w1</name>
      <index>2</index>
      <bandwidth>64000</bandwidth>
      <hardwareDescription>WAN 1 / IBM 2210</hardwareDescription>
      <functionalDescription>Enlace do POP-PB a UFPB/Patos (64
Kbps)</functionalDescription>
```

```
        <statusOID>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</statusOID>
        <allGraphics>
          <variableGraphic>
            <title>Status Operacional</title>
            <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</oid>
            <label>Fracao do tempo Up</label>
            <limit>1</limit>
          </variableGraphic>
          <variableGraphic>
            <title>Numero Total de Octets Recebidos</title>
            <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInOctets</oid>
            <label>Pacotes</label>
            <limit>64000</limit>
          </variableGraphic>
          <variableGraphic>
            <title>Numero Total de Octets Enviados</title>
            <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutOctets</oid>
            <label>Pacotes</label>
            <limit>64000</limit>
          </variableGraphic>
          <variableGraphic>
            <title>Erros nos Pacotes Recebidos</title>
            <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInErrors</oid>
            <label>Pacotes</label>
            <limit>100</limit>
          </variableGraphic>
          <variableGraphic>
            <title>Erros nos Pacotes Enviados</title>
            <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutErrors</oid>
            <label>Pacotes</label>
            <limit>100</limit>
          </variableGraphic>
        </allGraphics>
      </interface>
      <interface>
        <name>rt2210_w2</name>
        <index>3</index>
        <bandwidth>64000</bandwidth>
        <hardwareDescription>WAN 2 / IBM 2210</hardwareDescription>
        <functionalDescription>Enlace do POP-PB ao PAQTC/Cgrande (64
Kbps)</functionalDescription>
        <statusOID>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</statusOID>
        <allGraphics>
```

```
        <variableGraphic>
          <title>Status Operacional</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</oid>
          <label>Fracao do tempo Up</label>
          <limit>1</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Numero Total de Octets Recebidos</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInOctets</oid>
          <label>Pacotes</label>
          <limit>64000</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Numero Total de Octets Enviados</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutOctets</oid>
          <label>Pacotes</label>
          <limit>64000</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Recebidos</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInErrors</oid>
          <label>Pacotes</label>
          <limit>100</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Enviados</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutErrors</oid>
          <label>Pacotes</label>
          <limit>100</limit>
        </variableGraphic>
      </allGraphics>
    </interface>
    <interface>
      <name>rt2210_w3</name>
      <index>4</index>
      <bandwidth>64000</bandwidth>
      <hardwareDescription>WAN 3 / IBM 2210</hardwareDescription>
      <functionalDescription>Enlace do POP-PB a UFPB de Areia (64
Kbps)</functionalDescription>
      <statusOID>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</statusOID>
      <allGraphics>
        <variableGraphic>
          <title>Status Operacional</title>
```

```
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</oid>
          <label>Fracao do tempo Up</label>
          <limit>1</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Numero Total de Octets Recebidos</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInOctets</oid>
          <label>Pacotes</label>
          <limit>64000</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Numero Total de Octets Enviados</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutOctets</oid>
          <label>Pacotes</label>
          <limit>64000</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Recebidos</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInErrors</oid>
          <label>Pacotes</label>
          <limit>100</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Enviados</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutErrors</oid>
          <label>Pacotes</label>
          <limit>100</limit>
        </variableGraphic>
      </allGraphics>
    </interface>
    <interface>
      <name>rt2210_w4</name>
      <index>5</index>
      <bandwidth>1000000</bandwidth>
      <hardwareDescription>WAN 4 / IBM 2210</hardwareDescription>
      <functionalDescription>Enlace do POP-PB a UFPB de Jpessoa
(1Mbps)</functionalDescription>
      <statusOID>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</statusOID>
      <allGraphics>
        <variableGraphic>
          <title>Status Operacional</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOperStatus</oid>
          <label>Fracao do tempo Up</label>
```

```
          <limit>1</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Numero Total de Octets Recebidos</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInOctets</oid>
          <label>Pacotes</label>
          <limit>1000000</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Numero Total de Octets Enviados</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutOctets</oid>
          <label>Pacotes</label>
          <limit>1000000</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Recebidos</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifInErrors</oid>
          <label>Pacotes</label>
          <limit>100</limit>
        </variableGraphic>
        <variableGraphic>
          <title>Erros nos Pacotes Enviados</title>
          <oid>.iso.org.dod.internet.mgmt.mib-
2.interfaces.ifTable.ifEntry.ifOutErrors</oid>
          <label>Pacotes</label>
          <limit>100</limit>
        </variableGraphic>
      </allGraphics>
    </interface>
  </Interfaces>
 </NetEquipment>
</webMngrConfig>
```

Apêndice B

Configuração XML para os Mapas da Rede do POP-Pb

```
<?xml version="1.0" ?>
<Network>
  <title>POP-PB-v1</title>
  <description>Rede do POP-PB versao 1</description>
  <maps>
    <map id="main" height="520" width="550" backgroundColor="white" propagationLevel="8">
       <mapImage id="brasil" x="0" y="0" drawPlane="-1" imageFile="brasil.gif"/>
      <subMap id="poppb" x="83" y="30" imageFile="uppoint.gif" link="$poppb"/>
        <label value="POP-PB" x="74" y="32"/>
      <subMap id="poppe" x="91" y="35" imageFile="uppoint.gif" link="$poppe"/>
        <label value="POP-PE" x="74" y="32"/>
      <managedElement id="atmRJ" x="71" y="70" />
        <label value="POP-RJ" x="69" y="74"/>
      <managedElement id="atmSP" x="60" y="71" />
        <label value="POP-SP" x="58" y="75"/>
      <mapLine color="black">
          <sourceElem id="poppb"/>
            <targetElem id="atmPE"/>
        </mapLine>
      <mapLine color="black">
          <sourceElem id="atmPE"/>
            <targetElem id="atmSP"/>
        </mapLine>
      <mapLine color="black">
          <sourceElem id="atmPE"/>
            <targetElem id="atmRJ"/>
        </mapLine>
    </map>
    <map id="poppb" height="450" width="700" backgroundColor="white" propagationLevel="8">
      <subMap id="main" x="48" y="10" imageFile="nuvem.gif" link="$main"/>
        <label value="RNP-2 / INTERNET" x="45" y="7"/>
```

```
    <subMap id="rpp" x="70" y="70" imageFile="nuvem.gif" link="$rpp"/>
      <label value="RPP" x="73" y="80"/>
    <subMap id="proxy" x="23" y="70" imageFile="nuvem.gif" link="$proxy"/>
      <label value="Servico Web Proxy" x="20" y="80"/>
    <mapImage id="ciscoPB" x="33.5" y="35" imageFile="cisco2500.gif">
      <managedElement id="bb2" x="47.5" y="33"/>
      <label value="Roteador POP-PB" x="65" y="-10"/>
    </mapImage>
    <mapLine color="black">
        <sourceElem id="rnp"/>
          <targetElem id="ciscoPB"/>
      </mapLine>
    <managedElement id="bb2_s0" x="51.5" y="23"/>
      <mapLine color="black">
        <sourceElem id="rpp"/>
          <targetElem id="ciscoPB"/>
      </mapLine>
    <mapLine color="black">
        <sourceElem id="proxy"/>
          <targetElem id="ciscoPB"/>
    </mapLine>
  </map>
  <map id="poppe" height="450" width="700" backgroundColor="white" propagationLevel="8">
    <subMap id="main" x="45" y="15" imageFile="nuvem.gif" link="$main"/>
      <label value="RNP" x="46" y="-90"/>
    <mapImage id="img02" x="45" y="30" imageFile="nuvem.gif">
      <managedElement id="atmPE" x="40" y="40"/>
      <label value="POP-PE / ATM" x="110" y="30"/>
      <label value="(Recife/PE)" x="120" y="70"/>
    </mapImage>
    <mapImage id="img03" x="45" y="55" imageFile="nuvem.gif">
      <managedElement id="ciscoPE" x="40" y="40"/>
      <label value="POP-PE / DHC" x="-140" y="30"/>
      <label value="(Recife/PE)" x="-130" y="70"/>
    </mapImage>
    <subMap id="poppb" x="45" y="80" imageFile="nuvem.gif" link="$poppb"/>
      <label value="POP-PB" x="46" y="90"/>
      <label value="(Campina Grande/PB)" x="41" y="93"/>
    <mapLine color="black">
        <sourceElem id="main"/>
          <targetElem id="atmPE"/>
    </mapLine>
    <mapLine color="black">
        <sourceElem id="atmPE"/>
          <targetElem id="ciscoPE"/>
    </mapLine>
    <mapLine color="black">
        <sourceElem id="ciscoPE"/>
          <targetElem id="main"/>
      </mapLine>
```

```
</map>
<map id="rpp" height="370" width="565" backgroundColor="white" propagationLevel="8">
  <mapImage id="ibm2210" x="40" y="15" imageFile="ibm2210.gif">
    <managedElement id="rt2210" x="46" y="30"/>
    <label value="Roteador de Distribuicao" x="-10" y="-40"/>
    <label value="(IBM 2210)" x="20" y="-20"/>
  </mapImage>
  <mapImage id="ufpbCG" x="0" y="70" imageFile="nuvemgrande.gif">
    <label value="UFPB" x="35" y="50"/>
    <label value="C.Grande" x="24" y="65"/>
  </mapImage>
  <mapLine color="black">
      <managedElement id="sw8271_e3" x="54.5" y="40"/>
    <sourceElem id="ibm2210"/>
        <targetElem id="ufpbCG"/>
    </mapLine>
  <mapImage id="patos" x="20" y="70" imageFile="nuvemgrande.gif">
    <label value="UFPB" x="35" y="50"/>
    <label value="Patos" x="35" y="65"/>
  </mapImage>
  <mapLine color="black">
      <managedElement id="rt2210_w1" x="49.5" y="40"/>
    <sourceElem id="ibm2210"/>
        <targetElem id="patos"/>
    </mapLine>
  <mapImage id="paqtc" x="40.5" y="70" imageFile="nuvemgrande.gif">
    <label value="PaqTc" x="35" y="51"/>
  </mapImage>
  <mapLine color="black">
    <managedElement id="rt2210_w2" x="-550" y="40"/>
    <sourceElem id="ibm2210"/>
        <targetElem id="paqtc"/>
    </mapLine>
  <mapImage id="areia" x="60" y="70" imageFile="nuvemgrande.gif">
    <label value="UFPB" x="35" y="50"/>
    <label value="Areia" x="35" y="65"/>
  </mapImage>
  <mapLine color="black">
    <managedElement id="rt2210_w3" x="36" y="40"/>
    <sourceElem id="ibm2210"/>
        <targetElem id="areia"/>
    </mapLine>
  <mapImage id="ufpbJP" x="80" y="70" imageFile="nuvemgrande.gif">
    <label value="UFPB" x="35" y="50"/>
    <label value="J.Pessoa" x="24" y="65"/>
  </mapImage>
  <mapLine color="black">
    <managedElement id="rt2210_w4" x="41" y="40"/>
      <sourceElem id="ibm2210"/>
        <targetElem id="ufpbJP"/>
```

```
        </mapLine>
    </map>
    <map id="proxy" height="450" width="700" backgroundColor="white" propagationLevel="8">
      <mapImage id="ibm8271" x="40" y="15" imageFile="ibm8271.gif">
        <managedElement id="sw8271" x="46" y="30"/>
        <label value="Switch POP-PB.RNP.BR" x="10" y="-40"/>
        <label value="(IBM 8271)" x="30" y="-20"/>
      </mapImage>
      <mapImage id="estnd1" x="10" y="70" imageFile="estacao.gif">
        <managedElement id="nd1" x="0" y="0"/>
        <label value="ND 1" x="35" y="110"/>
      </mapImage>
      <mapLine color="black">
        <sourceElem id="ibm8271"/>
            <targetElem id="nd1"/>
        </mapLine>
      <mapImage id="estnd2" x="35" y="70" imageFile="estacao.gif">
        <managedElement id="nd2" x="0" y="0"/>
        <label value="ND 2" x="35" y="110"/>
      </mapImage>
      <mapLine color="black">
        <sourceElem id="ibm8271"/>
            <targetElem id="nd2"/>
        </mapLine>
      <mapImage id="proxy1" x="60" y="70" imageFile="estacao.gif">
        <managedElement id="proxy01" x="0" y="0"/>
        <label value="Proxy 1" x="25" y="110"/>
      </mapImage>
      <mapLine color="black">
        <sourceElem id="ibm8271"/>
            <targetElem id="proxy1"/>
        </mapLine>
      <mapImage id="proxy2" x="85" y="70" imageFile="estacao.gif">
        <managedElement id="proxy02" x="0" y="0"/>
        <label value="Proxy 2" x="25" y="110"/>
      </mapImage>
      <mapLine color="black">
        <sourceElem id="ibm8271"/>
            <targetElem id="proxy2"/>
        </mapLine>
    </map>
  </maps>
</Network>
```

Apêndice C

Diagramas de Classes Desenvolvidas

C.1 Pacote webmngr

C.2 Pacote webmngr.config

C.3 Pacote webmngr.monitor

C.4 Pacote webmngr.log

C.5 Pacote webmngr.status

C.6 Pacote webmngr.event

C.7 Pacote webmngr.info

C.8 Pacote webmngr.mail

C.9 Pacote webmngr.servlet

C.10 Pacote webmngr.util

Apêndice D

Glossário

API	Application Programming Interface
ASP	Active Server Page
ASN.1	Abstract Syntax Notation One
BER	Common Gateway Interface
CGI	Basic Enconding Rules
CORBA	Common Object Request Broker Architecture
COM	Component Object Model
DMI	Desktop Management Interface
DMTF	Distributed Management Task Force
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transport Protocol
HTTP-S	Secure Hyper Text Transport Protocol
IP	Internet Protocol
ISO	International Standards Organization
ITU-T	International Telecommunications Union
JVM	Java Virtual Machine
JMX	Java Management Extensions
JSP	Java Server Pages
MIB	Management Information Base
PDU	Protocol Data Unit
OO	Orientação a Objetos

OSI	Open System Interconnection
RMI	Remote Method Invocation
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SNMPv1	Simple Network Management Protocol version l
SNMPv2	Simple Network Management Protocol version 2
SNMPv3	Simple Network Management Protocol version 3
TCP	Transmission Control Protocol
URL	Uniform Resource Location
WBEM	Web-based Enterprise Management
XML	Extensible Markup Language