UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE FERRAMENTA PARA MONITORAÇÃO DE COMPUTADORES UTILIZANDO O PADRÃO DE GERÊNCIA WMI DA MICROSOFT E A PLATAFORMA DE DESENVOLVIMENTO .NET

RODRIGO JACOBOWSKI

BLUMENAU
2004

2004/2-43

RODRIGO JACOBOWSKI

PROTÓTIPO DE FERRAMENTA PARA MONITORAÇÃO DE COMPUTADORES UTILIZANDO O PADRÃO DE GERÊNCIA WMI DA MICROSOFT E A PLATAFORMA DE DESENVOLVIMENTO .NET

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Francisco Adell Péricas - Orientador

BLUMENAU
2004

2004/2-43

PROTÓTIPO DE FERRAMENTA PARA MONITORAÇÃO DE COMPUTADORES UTILIZANDO O PADRÃO DE GERÊNCIA WMI DA MICROSOFT E A PLATAFORMA DE DESENVOLVIMENTO .NET

Por

RODRIGO JACOBOWSKI

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Francisco Adell Péricas - Orientador, FURB
Membro:	Prof. Marcel Hugo, FURB
Membro:	Prof. Alexander Roberto Valdameri, FURB

Blumenau, 17 de Novembro de 2004

Dedico este trabalho a todos a todas as pessoas, que de alguma forma me ajudaram diretamente na realização deste.

Os bons livros fazem “sacar” para fora o que a pessoa tem de melhor dentro dela.

Lina Sotis Francesco Moratti

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À minha família, que mesmo longe, sempre esteve presente e em especial a meus pais e meu irmão e a noiva dele por sempre acreditarem em mim.

A minha namorada Dorinha que sempre me apoiou e esteve ao meu lado nos momentos bons e ruins.

Ao meu cachorro NIKI, que nos deixou no dia 10 de setembro após 14 anos em que estivemos juntos.

Ao meu orientador, Francisco Adell Péricas, que por sinal merece essa dedicatória, por ter acreditado na conclusão deste trabalho e ser realmente uma pessoa amiga e que se interessa pelo sucesso das outras.

Agradeço também a todos meus amigos que estiveram comigo nesta jornada.

RESUMO

Este trabalho apresenta um estudo sobre o padrão *Web Based Enterprise Management* (WBEM) da *Distributed Management Task Force* (DMTF) através da especificação e implementação de um protótipo de software de gerenciamento de sistemas, utilizando o *Windows Management Instrumentation* (WMI), o *framework* .NET e a linguagem de programação *Object Pascal*.

Palavras chaves: WBEM, WMI, .NET, Object Pascal.

ABSTRACT

This work presents a study about the *Web Based Enterprise Management* (WBEM) from *Distributed Management Task Force* (DMTF) through a specification and an implementation of a software prototype of a management system, using *Windows Management Instrumentation* (WMI), *framework* .NET and programming language *Object Pascal*.

Key-Words: WBEM, WMI, .NET, Object Pascal.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API – *Application Program Interface*

CIM – *Common Information Model*

CIMOM – *Common Information Model Object Manager*

CLR – *Common Language Runtime*

CLS – *Common Language Specification*

CMIP – *Common Management Information Protocol*

CMOT – *Common Management Information Protocol*

DCOM – *Distributed Component Object Model*

DMTF – *Distributed Management Task Force*

HMP – *Host Management Protocol*

HEMS – *High-level Entity Management System*

HTTP - *Hipertext Transfer Protocol*

IAB - *Internet Architecture Board*

IL – *Intermediate Language*

ISO – *International Organization for Standardization*

MIB – *Management Information Base*

MOF – *Managed Object Format*

MSIL – *Microsoft Intermediate Language*

OSI – *Open Systems InterConnection*

SGMP – *Simple Gateway Monitoring Protocol*

SNMP – *Simple Network Management Protocol*

SQL – *Structured Query Language*

TI – Tecnologia de Informação

UML – *Unified Modeling Language*

VNC – *Virtual Network Computing*

WBEM – *Web Based Enterprise Management*

WMI – *Windows Management Instrumentation*

WQL – *WMI Query Language*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, como o computador e a internet passaram a fazer parte de nosso dia a dia, viu-se a necessidade de poder-se ter um controle dos mesmos. Como a diversidade de informações encontradas na internet hoje é muita grande, não se pode deixar de ter controle sobre o que acontece com os computadores.

Os administradores de rede geralmente são pessoas ligadas à parte de informática na empresa, ou seja, são eles que permitem ou não alguma atividade de determinado usuário na rede.

Como os administradores de rede nem sempre possuem tempo disponível para controlar tudo o que acontece na rede, o surgimento do padrão de gerência *Windows Management Instrumentation* (WMI) possibilita manipular automaticamente as solicitações de clientes, conversar com os provedores e gerenciar o repositório de dados. Este serviço é disponibilizado aos programadores por intermédio de duas *Application Program Interfaces* (API), sendo que, para linguagem C/C++ é utilizado a API COM, que é o acesso de mais baixo nível ao WMI que se pode solicitar. Com isso surgiu a possibilidade do administrador desenvolver uma ferramenta capaz de monitorar a utilização de diversos computadores de uma rede interna.

Algumas destas informações, como o monitoramento do acesso a páginas da internet e a geração de arquivos de registro de atividades, já podem ser obtidas através de outras ferramentas disponibilizadas na internet como o *Virtual Network Computing* (VNC), porém não é gratuito.

O VNC tem como usabilidade o acesso remoto a outros computadores, com isto optou-se por implementar principalmente algumas funcionalidades não disponíveis na ferramenta VNC, como o monitoramento de instalação e remoção de software e hardware e envio de mensagens de *e-mail* para o administrador alertando para alguma eventual alteração de hardware/software.

1.1 OBJETIVOS DO TRABALHO

Este trabalho tem por objetivo desenvolver uma ferramenta de gerenciamento de computadores que auxilia o administrador da rede no monitoramento da instalação e remoção de software e hardware de determinado computador.

O trabalho possui os seguintes objetivos específicos:

a) visualizar informações de inventário de um computador conectado à rede;
b) efetuar comparativo entre arquivos de inventário para verificar alguma alteração que tenha ocorrido;
c) enviar *e-mail* ao administrador da rede notificando-o que alguma alteração ocorreu em algum dos computadores da rede.

1.2 ESTRUTURA DO TRABALHO

O capítulo 1 apresenta a estrutura geral do trabalho: introdução, objetivos, localização dos assuntos abordados e a organização do trabalho.

O capítulo 2 apresenta a gerencia corporativa, suas características, tipos, arquitetura, ares de funcionamento e os protocolos de gerenciamento.

O capítulo 3 apresenta o WBEM da DTMF e o WMI da Microsoft. Sobre o WBEM são abordados, conceitos, funcionalidades, arquitetura, componentes, modelo de dados e sobre os arquivos MOF, exemplificando um arquivo. Sobre o WMI são abordados suas características, arquitetura, funcionamento e a linguagem de consulta WQL.

O capítulo 4 apresenta o conceito de funcionamento da plataforma .NET, e a utilização do .NET *framework.*

O capítulo 5 apresenta o trabalho correlato ao qual este trabalho segue como uma extensão.

O capítulo 6 apresenta o desenvolvimento do trabalho, especificação, implementação e resultados obtidos com o desenvolvimento.

O capítulo 7 apresenta as conclusões e sugestões para a continuidade do trabalho.

2 GERÊNCIA CORPORATIVA

A gerência corporativa é definida como um conjunto de aplicações associadas a softwares e hardwares que tem como objetivo monitorar e controlar os recursos de informática de uma empresa (CARVILHE, 2000a). Segundo Carvilhe (2000a), a principal função dos sistemas de gerência corporativos é garantir a comunicação das informações da empresa de forma eficiente e segura, integrando aplicações para se obter o gerenciamento distribuído dos recursos de informática.

O maior desafio da gerência corporativa é o monitoramento e controle das diversas plataformas de hardware e software existentes e emergentes, somado à complexidade de gerenciamento dos mais diversos sistemas, aplicações e serviços existentes em uma rede heterogênea (CARVILHE, 2000a).

2.1 GERÊNCIA DE REDES

A gerência de redes é o conjunto de atividades voltadas para o planejamento, monitoramento e controle dos serviços prestados pela infra-estrutura de rede e pelas aplicações que dependem dessa infra-estrutura.

A gerência de redes procura maximizar o desempenho, aprovisionar recursos diante de alterações de demanda, minimizar falhas, documentar e manter configurações, além de zelar pela segurança dos elementos que compõem a rede.

O bom gerenciamento da rede faz com que os recursos sejam aproveitados da melhor forma possível, garantindo o retorno esperado para o investimento em tecnologia de informação (TI).

2.2 ARQUITETURA

A maioria das arquiteturas de gerenciamento de redes utiliza a mesma estrutura básica e conjunto de relações. Dispositivos gerenciáveis, tais como computadores ou dispositivos de rede, executam um software agente que os habilita a enviar alertas quando algum problema é detectado. Recebendo esses alertas, as entidades de gerenciamento são programadas para reagir executando uma ou várias ações, incluindo notificação aos operadores do sistema, inclusão de eventos ao histórico, desligamento do dispositivo, e tentativa de reparo automático.

Entidades de gerenciamento também podem requisitar valores de certas variáveis às estações da rede. Essas requisições podem ser automáticas ou ativadas pelo usuário, mas o agente no dispositivo gerenciado responde a todas as requisições. A figura 1 mostra uma típica arquitetura de gerenciamento de rede.

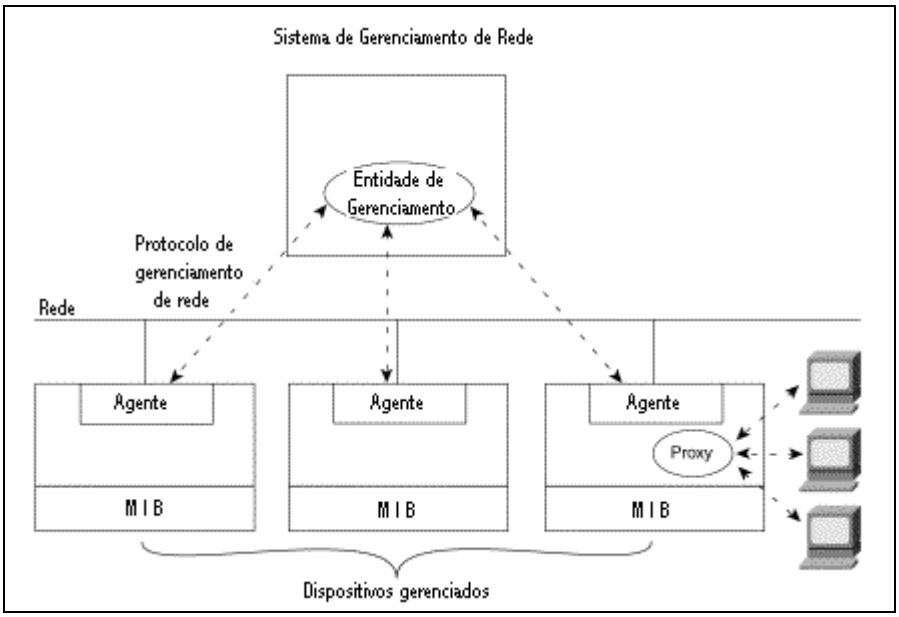


Figura 1 - Arquitetura de gerenciamento de rede

2.3 ÁREAS FUNCIONAIS

Segundo a *International Organization for Standardization* (ISO), uma organização internacional de padronização, as diversas atividades de gerenciamento de redes podem ser divididas em cinco áreas funcionais específicas definidas na figura 2, denominadas de gerenciamento de falhas, gerenciamento de desempenho, gerenciamento de configuração, gerenciamento de contabilização e gerenciamento de segurança (CARVILHE, 2000a).

A seguir, são caracterizadas as áreas funcionais de gerenciamento de redes propostas pela ISO:

a) gerência de falhas: corresponde à área funcional que permite a detecção, o isolamento e a correção de operações anormais na rede. Os recursos de gerenciamento de falhas mostram ao administrador de rede, o número, tipos, hora de ocorrência e localizações de erros na rede. Quando ocorrem falhas em uma rede, é importante que os seguintes procedimentos sejam seguidos: localizar a falha, isolar a falha do restante da rede e reparar os componentes em falha, de forma a retornar a rede ao seu estado normal;

b) gerência de desempenho: os elementos que compõem uma rede precisam ser monitorados de forma constante com a finalidade de avaliar o seu comportamento. Tais informações podem ser utilizadas para fins de planejamento e controle da qualidade de serviço na rede. Através de estatísticas de desempenho, podem–se promover ações para antecipar-se a problemas que venham a ocorrer pela degradação crescente dos tempos de resposta, motivado por problemas ou por saturação de capacidade dos equipamentos ou dispositivos na rede. O sistema deve prover o estabelecimento de limiares de comportamento permitidos para cada elemento, de forma que sejam emitidas notificações para motivar uma ação quando estes valores forem atingidos;

c) gerência de configuração: compreende o conjunto de facilidades que lidam com a instalação, inicialização, modificação e registro de parâmetros de configuração. As redes de computadores podem constituir-se de milhares de equipamentos e dispositivos dispersos por vários locais físicos diferentes em uma organização, muitos deles envolvidos em mudanças freqüentes de localização. Para a gerência da rede é fundamental que sejam conhecidas a localização de cada equipamento ou dispositivo, suas especificações técnicas e configurações, os responsáveis pela manutenção ou correção de problemas, dentre outras informações;

d) gerência de contabilização: registra as informações sobre a utilização dos recursos da rede com o objetivo de quantificá-los para efeito de distribuição de custos, de tarifação, de planejamento de capacidade, e de verificação de cotas de utilização;

e) gerência de segurança: corresponde ao conjunto de funções responsáveis pela criação e supressão de mecanismos de segurança na rede. Uma rede de computadores de uma organização somente deve ser acessada por pessoas ou aplicações que possuam a devida autorização. As informações sensíveis para o negócio da corporação devem ser mantidas de forma segura, prevenindo-se contra os acessos indevidos, seja para leitura ou para alteração da informação. Os procedimentos de uma gerência de segurança devem incluir a identificação dos pontos de acesso em uma rede, definição dos procedimentos de segurança e, principalmente, manter estes pontos seguros, informando inclusive as tentativas de ataque para uma ação preventiva. Estão inseridos neste contexto os procedimentos de autenticação dos usuários para acesso aos sistemas de processamento, a implementação de *firewalls*, as técnicas de criptografia para o transporte da informação, a geração e manutenção de cópias de segurança dos arquivos dentre outros.

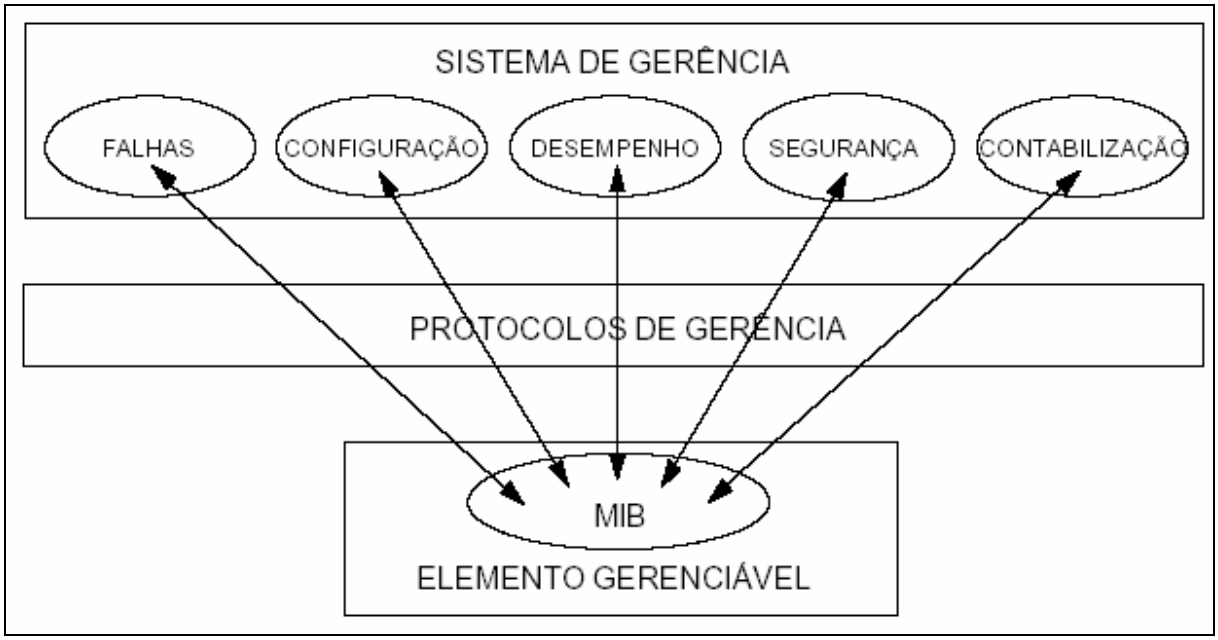


Figura 2: Áreas funcionais no sistema de gerência de rede

2.4 PROTOCOLOS DE GERENCIAMENTO DE REDES

O primeiro dos protocolos de gerência de rede foi o *Simple Gateway Monitoring Protocol* (SGMP) que surgiu em novembro de 1987. Entretanto, o SGMP era restrito à monitoração de *gateways*. A necessidade crescente de uma ferramenta de gerenciamento de rede mais genérica fez emergirem mais algumas abordagens (SPECIALSKI 2002, p.12):

a) *High-level Entity Management System* (HEMS), generalização do *Host Management Protocol* (HMP);
b) *Simple Network Management Protocol* (SNMP), um melhoramento do SGMP;
c) *Common Management Information Protocol* (CMOT) *over* TCP/IP uma tentativa de incorporar o máximo possível o protocolo *Common Management Information Protocol* (CMIP), serviços e estrutura de base de dados que estava sendo padronizada pela ISO para gerenciamento de redes.

No início de 1988, a *Internet Architecture Board* (IAB) revisou os protocolos e escolheu o SNMP como uma solução de curto prazo e o CMIP como solução de longo prazo para o gerenciamento de redes. O sentimento era que, em um período de tempo razoável, as instalações migrariam do TCP/IP para protocolos baseados em *Open Systems InterConnection* (OSI). Entretanto, como a padronização do gerenciamento baseado no modelo OSI apresentava muita complexidade de implementação e o SNMP, devido à sua simplicidade, foi amplamente implementado nos produtos comerciais, o SNMP tornou-se um padrão de fato.

Posteriormente, pela existência de lacunas funcionais (devido exatamente à simplicidade do SNMP), foram definidas novas versões do protocolo SNMP chamadas de SNMPv2 e SNMPv3, e o SNMP original ficou conhecido como SNMPv1 (SPECIALSKI 2002, p.14).

A primeira versão da arquitetura de gerenciamento SNMP foi definida no RFC 1157 de maio de 1990 (SPECIALSKI 2002, p.14).

O RFC 1157 define que a arquitetura SNMP consiste de uma solução para o problema de gerenciamento de redes, em termos de:

a) o escopo da informação de gerenciamento comunicada pelo protocolo;
b) a representação da informação de gerenciamento comunicada pelo protocolo;
c) operações sobre a informação de gerenciamento, suportadas pelo protocolo;
d) a forma e o significado das trocas entre entidades de gerenciamento;
e) a definição dos relacionamentos administrativos entre entidades de gerenciamento;
f) a forma e o significado das referências às informações de gerenciamento.

O RFC 1157 define ainda três objetivos a serem alcançados pelo SNMP: minimizar o número e complexidade das funções de gerenciamento, ser flexível o suficiente para permitir expansões futuras e ser independente da arquitetura e mecanismo dos dispositivos gerenciados.

3 WBEM

O *Web-Based Enterprise Management* (WBEM) é o resultado de uma iniciativa da indústria da informática para criar uma tecnologia padrão de gestão de informação num ambiente empresarial. Esta iniciativa foi patrocinada por algumas das maiores empresas do mundo na área, como são os casos da IBM, Compaq/HP, Cisco Systems e Microsoft.

Um dos principais objetivos desta iniciativa é a definição de um modelo de dados padrão que possa ser utilizado como repositório de dados de gerência para substituir *Management Information Bases* (MIBs) dos tradicionais protocolos. O esquema de dados do WBEM é totalmente orientado a objetos o que permite a herança de objetos gerenciados, abstração de dados, polimorfismo e encapsulamento, permitindo também que os objetos de sistemas legados ou proprietários sejam importados para o banco de dados de objetos de gerência WBEM, criando uma extensão desse esquema. Este modelo de dados é conhecido como *Common Information Model* (CIM) (DTMF, 1999). Com este modelo, a arquitetura torna-se um ponto de integração dos diversos *frameworks* existentes (CARVILHE, 2000b). Outro objetivo é o estabelecimento de um protocolo e uma linguagem padrão que irão permitir que os objetos gerenciados WBEM sejam acessados via rede. O ideal é que este protocolo e esta linguagem sejam independentes de plataforma e trabalhem sobre a internet.

A figura 3 representa o fluxo de dados básico do modelo WBEM dando uma visão geral e simplificada da forma como se dá seu funcionamento. Nela está representada a aplicação gerente que interage com o gerente de objetos chamado *Common Information Model Object Manager* (CIMOM), responsável pelo controle de objetos gerenciados incluindo o armazenamento e o acesso a objetos no repositório central de dados (CIM). O CIMOM possui a sua própria linguagem para definição de objetos gerenciados denominada *Managed Object Format* (MOF) (MOORE et.al. 2001).

A arquitetura WBEM define a estrutura e convenções necessárias para acessar informações sobre objetos gerenciados. Estes objetos são representados por classes, com seus atributos e métodos apropriados, sendo compatível com a maioria dos protocolos de gerenciamento tradicionais, como, por exemplo, o SNMP. Ultimamente esta padronização permitiu aos administradores o gerenciamento de *desktops*, dispositivos e redes a partir de um *web browser*.

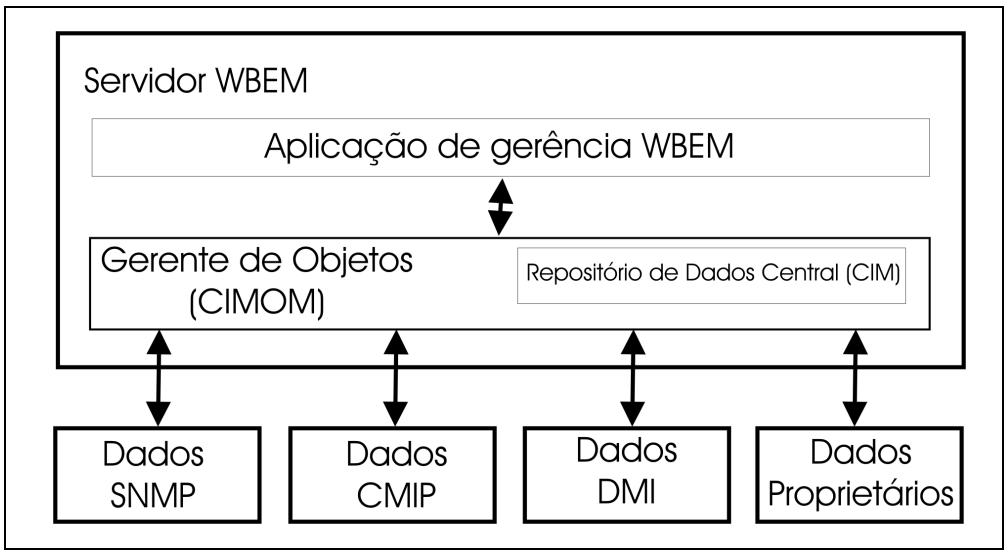
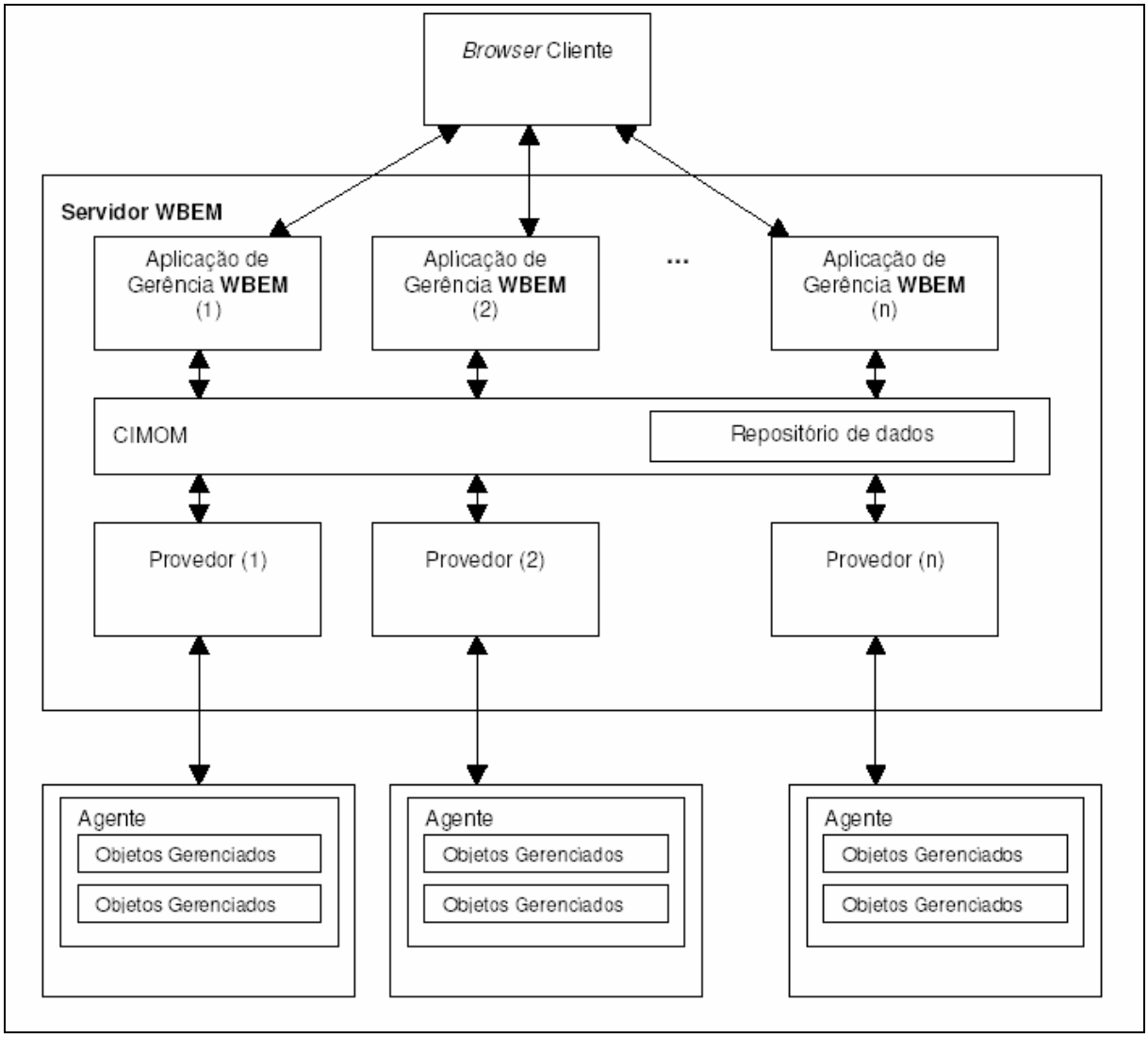


Figura 3: Modelo de fluxo de dados WBEM

Outro fator que é característica marcante deste modelo é a independência de padrão proposta. Isto significa que o modelo WBEM pode ser utilizado sem que seja necessária uma reestruturação dos protocolos já existentes, além de prover um gerenciamento baseado na orientação a objetos promovendo a integração com os dados dos objetos gerenciados o que torna independente de uma *Application Program Interface* (API) especificada (DMTF, 1999).

Os componentes representados na figura 4 apresentam o padrão WBEM mais completo. Nesta figura há o servidor WBEM, no qual está contido o CIMOM, que pode aceitar todos os objetos dos *frameworks* de gerência de sistemas e de redes existentes e onde os dados dos objetos são transformados em um formato padrão e armazenados no CIM, além de prover a aplicação WBEM, que por sua vez pode ser acessada por clientes WBEM através do *browser* cliente acessando as aplicações de gerência através de um protocolo. Quando executadas, as aplicações de gerência WBEM acessam o CIMOM, este por sua vez interpreta a requisição feita pela aplicação sobre um determinado objeto gerenciado e consulta ou atualiza tal informação no CIM. Da mesma forma, quando algum valor é alterado no objeto, a informação é entregue ao CIMOM pelos provedores, que correspondem a um conjunto de processos que se comunicam com os agentes de gerência de sistemas obtendo o valor dos objetos definidos para serem colocados no CIM. Vale frisar que apenas os dados estáticos são armazenados no CIM enquanto que os dinâmicos são consultados diretamente pelo CIMOM aos provedores.



Fonte: Braz (2003, p. 22)

Figura 4: Modelo WBEM

Para sintetizar melhor a funcionalidade dos provedores pode-se dizer que eles são responsáveis pela obtenção e apresentação dos dados de gerência dos protocolos convencionais de gerência de sistemas e de rede. O WBEM define basicamente 4 tipos de provedores sendo eles:

a) provedores de propriedades, que retornam os valores de propriedades de objetos através de uma chave de identificação;
b) provedores de instância, que retornam os valores de instância dos objetos;
c) provedores de classes, que retornam classes e instâncias;
d) provedores de *namespaces*, que são unidades de grupos de classes e instâncias que controlam a visibilidade das mesmas e não são localizações físicas, são apenas

referencias lógicas, e seus provedores são capazes de gerenciar um espaço de nome definido pelo CIMOM (CARVILHE, 2000b).

3.1 COMPONENTES DA ARQUITETURA WBEM

A arquitetura WBEM é formada pelo CIM (*CIM MetaSchema, CIM Core Model e CIM Commom Model*) e pelas extensões do esquema que englobam o MOF e o CIMOM, todos eles com suas especificações definida pela DMTF (1999).

3.1.1 CIM

O CIM define o modelo utilizado para representar os objetos gerenciados do mundo real através do paradigma de orientação a objetos. Conforme Carvilhe (2000a), o CIM foi projetado para receber informações de agentes, permitindo que aplicações corporativas de diferentes desenvolvedores que utilizam plataformas heterogêneas, descrevam, criem e compartilhem todos os objetos de gerência.

O CIM é composto por um meta esquema (CIM *Meta Schema*) e por esquemas padrões (CIM *Schemas*). Existe ainda um arquivo texto ASCII que utiliza a linguagem MOF, que contém as definições dos esquemas padrões implementados (CARVILHE, 2000b).

3.1.1.1 CIM META SCHEMA

O meta esquema corresponde à definição formal do modelo. Define os termos usados para sua modelagem, seu uso e semântica. Os principais elementos são (BRAZ 2003, p.24):

a) classe: é definida como o conjunto de instâncias do mesmo tipo, com as mesmas propriedades e mesmos métodos. É a unidade básica de definição da estrutura de gerenciamento;
b) evento ou *trigger*: é definido como a operação invocada pela alteração de algum estado, como criação de um objeto, alteração de uma propriedade de um objeto;
c) indicação: é definido como subtipos de classes ou seja, são objetos criados como resultado de um evento, que podem ter propriedades e métodos, e ser organizados seguindo uma hierarquia, ou seja, uma indicação é um tipo de classe;
d) método: é definido como uma operação que descreve o comportamento e o conteúdo que pode ser aplicado sobre uma classe;

e) propriedades: é definido como o valor para indicar as características de uma classe. A propriedade poder ser vista como as funções *get* e *set* que quando aplicadas ao objeto retornam e definem estados deste objeto;

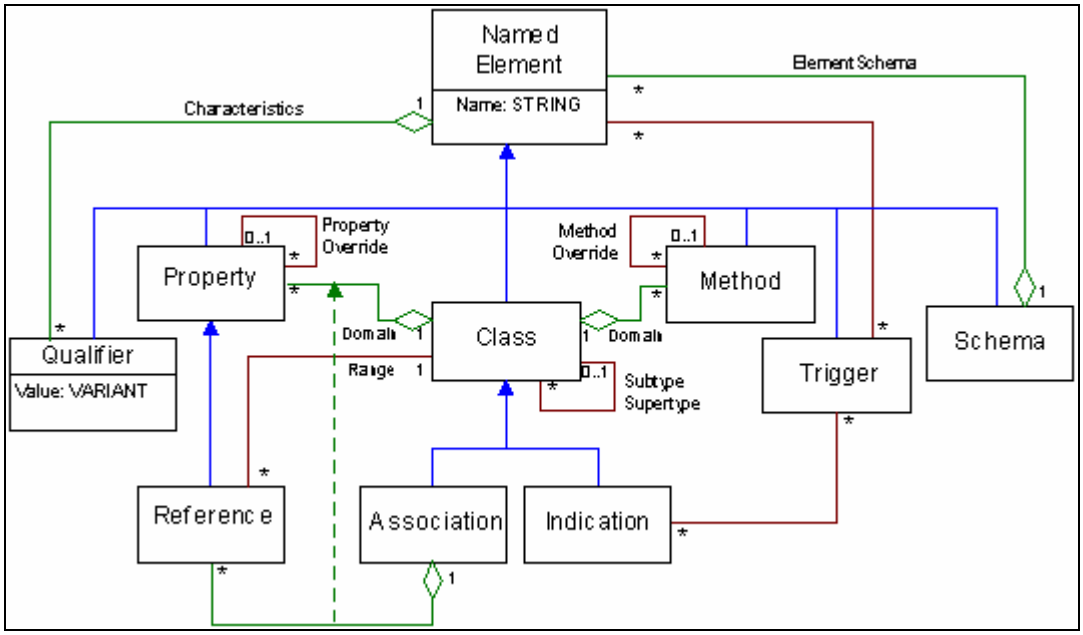
f) referência: são propriedades que uma classe ou instância possui. O valor é um ponteiro para um objeto;

g) associação: é um tipo de classe que possui uma ou mais referências. São utilizadas para criar relacionamentos entre objetos sem necessidade de alterar as suas definições;

h) qualificador: é utilizado para caracterizar classes, instâncias ou métodos. Permite a extensão do esquema de forma limitada e controlada. Por exemplo, existem qualificadores para definir as características de uma propriedade ou de uma chave de uma classe;

i) esquema: é um conjunto de classes com um único proprietário. Dentro de um esquema os nomes de classes que devem ser únicos.

A figura 5 demonstra a estrutura do meta esquema. O meta esquema completo é definido em um arquivo MOF e segue diversas regras detalhadas pelo DMTF (1999).

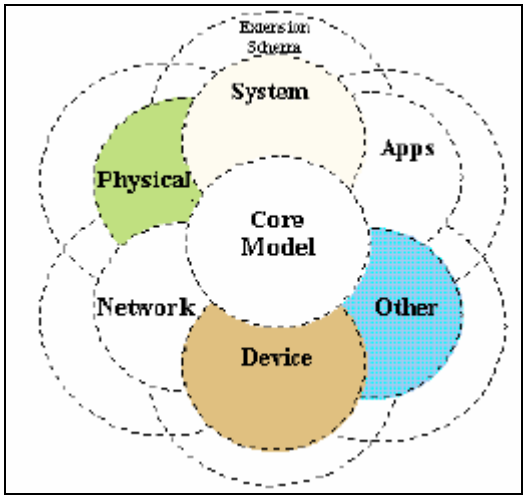


Fonte: DMTF (2002)

Figura 5 – Estrutura do CIM Meta Schema

3.1.1.2 CIM SCHEMA

Segundo Carvilhe (2000b), o esquema padrão corresponde a um conjunto padrão de classes com suas propriedades e associações. Estas propriedades e associações fornecem a base necessária para a organização que irá utilizar o padrão WBEM. O esquema padrão foi definido com o objetivo de prover a uniformidade dos procedimentos de requisição sobre a rede de gerenciamento. O esquema padrão (CIM Schema) é composto pelo *Core Model*, *Common Model* e *Extension Schemas.* A figura 6 representa as diversas camadas do esquema padrão.



Fonte: Braz (2003, p. 25)

Figura 6 – Camadas do CIM Schema

3.1.1.2.1 Core Model

Corresponde a um conjunto padrão de classes com suas propriedades e associações que fornecem a base necessária para a organização que irá utilizar o padrão WBEM, definido com o objetivo de promover a uniformidade dos procedimentos de requisição sobre o ambiente de gerenciamento. É constituído por um conjunto de objetos que são referenciados por todas as áreas de gerência (CARVILHE, 2000b). Portanto, o *core model* é um conjunto relativamente pequeno de classes, associações e propriedades que provêem o vocabulário básico para análise e descrição de sistemas gerenciados e é ponto inicial para o estudo de como determinar as extensões para domínios específicos (MICROSOFT, 1999).

3.1.1.2.2 Commom Model

Modela as informações de uma área de gerenciamento em particular, mas independente de plataforma, tecnologia ou implementação. As áreas deste modelo são:

a) sistemas (systems), que refere-se por exemplo aos vários tipos de sistemas de computadores;
b) aplicações (application), que modela as informações que descrevem e que geralmente são usadas e requeridas por software de gerência;
c) redes (networks), que representa os vários modelos de rede incluindo topologias, conectividade, e acesso fornecidos por vários protocolos e serviços necessários para prover acesso de rede (MICROSOFT, 1999);
d) dispositivos (devices), que contêm as classes que representam os dispositivos que integram os componentes físicos do sistema.

3.2 CIMOM

O *CIM Object Manager* (CIMOM) é um gerenciador de objetos que contém um modelo de dados cuja tarefa é consolidar e traduzir os dados de gerência provenientes de diferentes fontes conforme demonstra a figura 7.

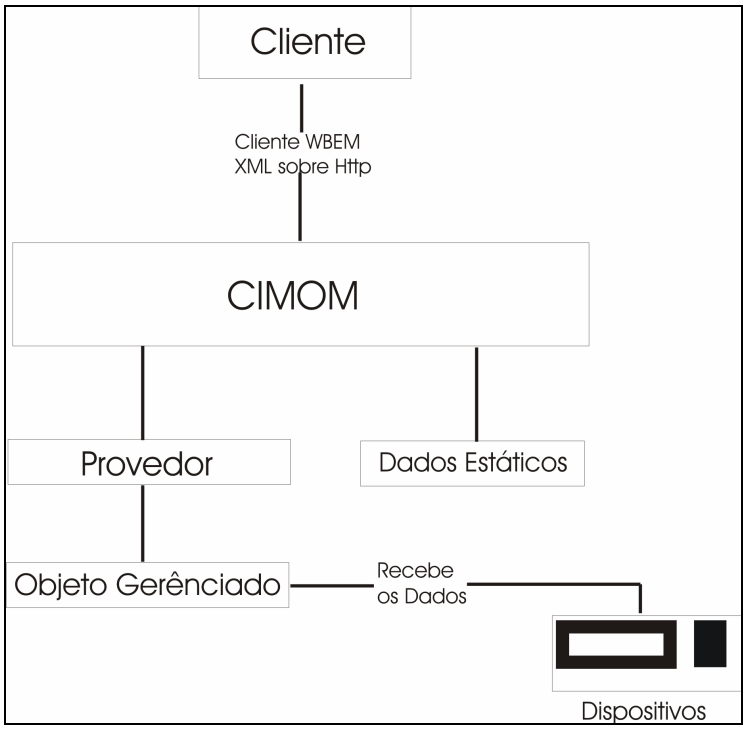


Figura 7 – CIMOM

Pode-se observar na figura 7, o cliente comunica com o CIMOM utilizando a linguagem XML sobre o protocolo HTTP. Um dos últimos estudos realizados pela iniciativa WBEM é a utilização da linguagem XML como método de representação das informações de gerência. Existe também a proposta do mapeamento entre o modelo CIM e construções XML.

O maior benefício disto é a padronização da linguagem XML para representação dos objetos de gerência. Estes dados poderiam desta forma ser apresentados e manipulados por aplicações de gerência corporativa WBEM.

3.3 MOF

O MOF, especificado pela DMTF, serve de entrada para o compilador MOF de aplicações de gerência, que tem o objetivo de criar chamadas apropriadas para o CIMOM. O compilador MOF define classes, propriedades e qualificadores para serem incluídos no banco de dados CIM, e também coloca a instância dessas classes neste mesmo banco de dados.

O conteúdo do arquivo MOF é carregado para o *namespaces* (espaço de nomes) que fornece o domínio no qual as instâncias de classes são identificadas por uma chave única que define o qualificador. Os *namespaces* são utilizados para definir bloco de gerenciamento para limitar o tamanho do banco de dados, definir visões do modelo somente para objetos de gerenciamento específicos e para pré estruturar os grupos de objetos para otimizar a performance das consultas (DMTF, 1999).

O arquivo MOF contém as definições de instâncias, classes ou ambos, como ilustra a figura 8, onde pode-se observar que um arquivo MOF pode apenas ser compilado e adicionado ao *namespaces*, assim como compilado e importado por outro *namespaces* ou simplesmente importado e adicionado ao *namespaces* pois já foi previamente compilado. No quadro 1 pode-se observar um exemplo de arquivo MOF.

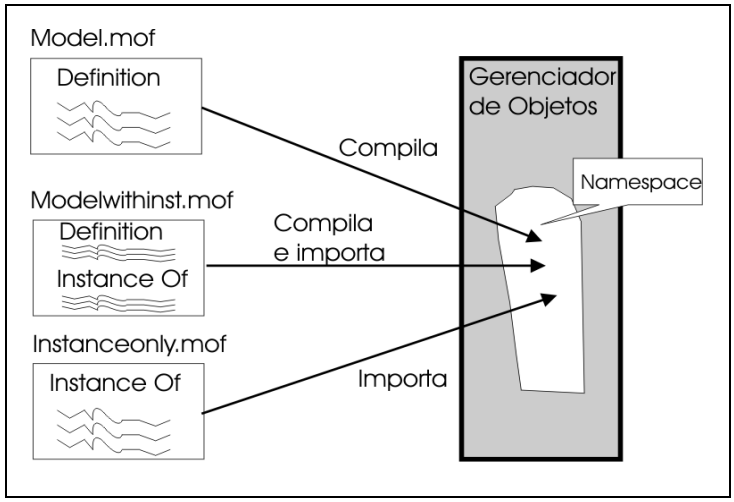


Figura 8 – Modelo MOF

```
//************************************************************
//* Class: Win32_WordTemplate
//* Derived from:
//************************************************************
[dynamic: ToInstance, provider("OffProv")]
class Win32_WordTemplate
{
      [key, read: ToInstance ToSubClass] string Name;
      [read: ToInstance ToSubClass] string Path;
      [read: ToInstance ToSubClass] string Type;

};
```

Quadro 1 – Arquivo MOF

3.4 WMI

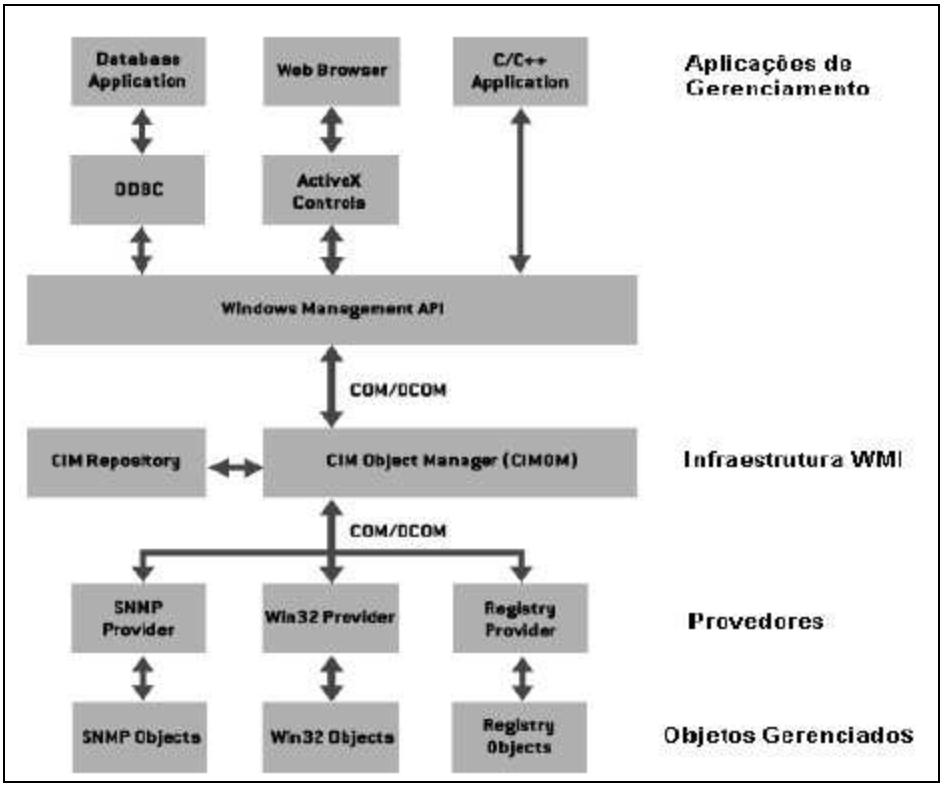
O *Windows Management Instrumentation* (WMI) é a implementação da Microsoft para o WBEM, que tem como objetivo estabelecer padrões para acessar e compartilhar informações de gerenciamento em uma rede corporativa podendo controlar e monitorar os componentes do sistema, sejam software ou hardware, além de fornecer suporte integrado ao CIM. O WMI inclui um repositório de dados compatível com o CIM e o gerenciador de objetos CIMOM (MICROSOFT, 2002).

3.4.1 ARQUITETURA WMI

A arquitetura do WMI consiste de quatro segmentos que são (MICROSOFT, 1999):

a) aplicações de gerenciamento, que podem acessar, exibir e processar dados obtidos de objetos gerenciados;
b) infra-estrutura de WMI, que é o elo de ligação entre aplicações de gerenciamento e provedores;
c) provedores, que tem como objetivo fornecedor dados e definições de classes, atuando como intermediários entre componentes do sistema operacional e aplicativos;
d) objetos gerenciados referentes a todos dispositivos de rede que possam ser gerenciados.

A arquitetura WMI é ilustrada na figura 9.



Fonte: Braz (2003, p. 33)

Figura 9 – Arquitetura WMI

O processo executável que provê todas as funcionalidades do WMI chama-se *WINMmt.exe.* O serviço WMI (*WinMmt.exe)* é responsável por manipular solicitações de clientes, conversar com os provedores e gerenciar o repositório de dados. Este serviço é disponibilizado aos programadores por intermédio de duas API, sendo que, para a linguagem C/C++ é utilizada a API COM, que é o acesso de mais baixo nível ao WMI que se pode solicitar. Já para linguagem de *scripts* que reconhecem automação, é utilizada a API *Scriptin*, que é um empacotador (*wrapper*) em torno da API COM (MARTINSSON, 2002, p.126).

O WMI também é implementado seguindo os conceitos de *namespaces*, onde uma aplicação de administração deve conectar-se a um *namespaces*, para poder ter acesso às propriedades de todos os objetos naquele *namespaces* (MICROSOFT, 2000). A sua segurança pode ser implementada com base nestes espaços de nomes, onde um administrador pode controlar quais usuários podem ter acesso a um determinado *namespaces*.

O WMI tem como principal limitação o fato de utilizar a comunicação *Distributed Component Object Model* (DCOM) entre o CIMOM e os provedores e entre o CIMOM e a API de gerenciamento, fugindo assim da proposta WBEM de utilizar *web browser* para enviar e solicitar informações trafegando-as sobre o protocolo HTTP utilizando a linguagem XML

independente de plataforma. Além do que, seria necessário uma implementação mais detalhada para se utilizar, por exemplo, uma aplicação de gerenciamento a partir de uma estação que se encontra atrás de um *firewall* ou passando por vários roteadores.

3.4.2 WMI QUERY LANGUAGE

A *WMI Query Language (WQL*) é uma linguagem de consultas ao WMI. Tem como objetivo enumerar, explorar e gerar relatórios sobre o andamento de um ambiente gerenciado (BRAZ 2003, p.35). A WQL é um subconjunto do *Structured Query Language* (SQL). Diferentemente do SQL tradicional, a WQL é uma linguagem de consultas que não foi concebida para atualizar, eliminar ou inserir dados. A WQL, segundo Braz Junior (2003), suporta três tipos de consultas:

a) consultas de dados: usada pelos aplicativos de gerenciamento que têm de selecionar associações de dados e instâncias sobre instâncias;
b) consultas de evento: usado pelos aplicativos de gerenciamento que implementam manipulação de eventos. Os eventos são disparados dentro de seu aplicativo de gerenciamento para notificá-lo de alterações como a criação, a eliminação ou a modificação de um banco de dados. Essas consultas registram o aplicativo de gerenciamento em termos de notificações de eventos;
c) consultas de esquema: similar às consultas de dados, porém estas retornam meta informações em vez de instâncias.

Uma consulta que permite selecionar todos os campos das instâncias da classe disco rígido que tenham espaço livre inferior a 50 Mb, utilizando a linguagem WQL pode ser visualizada no quadro 2.

```
Select * from Win32_PhysicalDisk where freeSpace<50000
```

Quadro 2 – Exemplo de consulta WQL

Informações detalhadas sobre a construção de consultas utilizando a WQL podem ser encontradas em Martinsson (2002, p.189).

4 PLATAFORMA .NET

A plataforma .NET é um ambiente de desenvolvimento poderoso, que permite o desenvolvimento de aplicações para *desktop* (para Windows ou console), aplicações para aparelhos móveis (*palm-tops*, celulares) e desenvolvimento de aplicações *web* (através da tecnologia ASP.NET).

A plataforma .NET tem a proposta de proporcionar um ambiente de desenvolvimento avançado, disponibilizando recursos poderosos para uso dos desenvolvedores. Para isso, a Microsoft unificou todas as soluções de desenvolvimento dela nessa nova plataforma, além de melhorar bastante os recursos oferecidos. Pode-se dizer que o .NET *framework* disponibiliza um ambiente de desenvolvimento multi-plataforma (em relação ao sistema operacional), multi-linguagem, orientado a objeto, e com uma grande e eficiente biblioteca de classes, como segue(D´ANGELO, 2003):

a) multi-plataforma: com um conceito similar à tecnologia JAVA, todo código desenvolvido, ao ser compilado, é interpretado, depurado (já contendo as verificações de lógica) e transformado em linguagem intermediária chamada *Microsoft Intermediate Language* (MSIL). Essa linguagem intermediária somente é entendida pelo *Common Language Runtime* (CLR). Quando um programa é executado pela primeira vez, a CLR lê o código MSIL e o transforma em linguagem de máquina (0 / 1), o qual é interpretado pelo processador. Não se pode dizer, no entanto, que a CLR interpreta o MSIL. A interpretação é feita quando o código escrito é compilado, como descrito acima. Dessa forma, o papel da CLR, é transferir o código MSIL para a linguagem nativa da máquina em questão. Assim sendo, toda aplicação construída no .NET *framework* pode ser executado em todas as plataformas que tem CLRs desenvolvidas. No entanto, só existe CLR homologada para a plataforma Windows até o momento;
b) multi-linguagem: no item anterior, foi citado que uma vez escrito o código, este deveria ser compilado e transformado em MSIL. Mas, em momento nenhum, foi citada em qual linguagem de programação este código deveria ser escrito. Isto porque qualquer linguagem que seja compatível com a plataforma de desenvolvimento .NET pode ser utilizada, ou seja, se um determinado compilador de linguagem segue as especificações da *Common Language Specification* (CLS), ela é compatível com .NET, e gera código MSIL compatível com a CLR. Hoje,

existem várias linguagens homologadas seguindo a CLS. Pode-se citar Delphi 8, C#, VB.NET e J#. Pode-se dizer então que um aplicativo “HELLO WORLD” escrito em qualquer umas dessas linguagens citadas, ao ser compilado, tem o mesmo código MSIL;

c) Orientação a Objetos: A plataforma de desenvolvimento .NET dá suporte total a orientação a objetos. Dessa forma, todas as linguagens homologadas devem ter suporte a orientação a objetos, e assim, permitir a criação de classes com propriedades e métodos, incluindo métodos construtor, herança, polimorfismo, agregação, sobrescrita de métodos, sobrecarga de métodos, entre outras. Deve ainda conter ferramenta de tratamento de erros. Dá suporte também à interoperabilidade das linguagens, ou seja, classes escritas em uma determinada linguagem, e compilada, pode ser lida por qualquer uma das outras linguagens que seguem as especificações da CLS, ou seja, é possível acessar, por exemplo, em um aplicativo escrito em C#, classes desenvolvidas em VB.NET.

Para completar, a plataforma de desenvolvimento .NET contém uma biblioteca de classes nativas, que incluem classes de acesso e manipulação de dados, classes que contém objetos visuais (*Windows forms*), classes de manipulação de informação transmitidas pela *web* (ASP.NET), classes de acesso ao sistema, entre outros. É uma infinidade de classes que têm como objetivo proporcionar ao desenvolvedor que o mesmo tenha o foco no negócio da aplicação (no objetivo da aplicação) e não na ampliação dos limites tecnológicos da plataforma de desenvolvimento utilizada.

4.1 .NET FRAMEWORK

A .NET *framework* é um sistema operacional dentro de outro, ou seja, tudo que é executado dentro dela é gerenciada por ela, o pedido de acesso à memória é feito a ela, na requisição de hardware o pedido é feito a ela, onde então ela lida com o sistema operacional, seja ele Windows, Linux, etc. Com isso pode-se fazer coisas que antes eram complexas usando ponteiros ou rotinas em linguagem de máquina. Todos os recursos anteriores disponíveis com relação a hardware continuam a estar disponíveis e, além disso, a junção de sistema local com sistema *web* se torna cada vez mais evidente, tornando possível, por exemplo, via *browser* controlar um aplicativo remotamente. Poder-se-ia, por exemplo, ao

clicar em um botão de um formulário dar *boot* na máquina (pelo menos conceitualmente falando).

A .NET *framework* é composta de *assemblies* que dão suporte a qualquer linguagem que execute dentro dela, tornando seu desenvolvimento mais fácil. Pode-se comparar os *assemblies* do .NET com os *packages* de *run-time* do Delphi, pois estão sempre em memória, e tornam o aplicativo final menor. Os *assemblies* são os arquivos executáveis .NET, onde estão contidas as funcionalidades. Eles são compilados para a forma de uma linguagem intermediária (LI). Esta LI é o padrão de linguagem que o .NET utiliza para ler os arquivos e depois executá-los. O *framework* é uma biblioteca de classes que espelham as funcionalidades do sistema operacional, ou seja, acabando com a penúria de programadores Windows ao ter que usar chamadas a API do Windows, tornando as tarefas muito mais simples e também tornando uma forma padronizada de se chamar rotinas do sistema operacional.

O .NET *framework* tem sua arquitetura aberta, permitindo a integração de outras empresas de software, tornando-se uma característica marcante sua capacidade de ter diversas linguagens interagindo entre elas inclusive Java, ou seja, pode-se criar uma classe em C#, herdá-la em VB.NET e depois usá-la em um software em Delphi. Tamanha é esta capacidade que já existem mais de 20 linguagens (Java, Delphi, Perl, Cobol, Python, Smalltalk, RPG, APL, Fortran, etc...) sendo portadas para o .NET. Muitas empresas estão apostando nesta plataforma e a tendência é que ela cresça cada vez mais (IMASTER, 2003).

4.2 TRABALHOS CORRELATOS

No desenvolvimento deste trabalho optou-se por desenvolver algumas extensões que foram sugeridas no trabalho de Braz (2003). No trabalho desenvolvido por Braz (2003), foi construído um protótipo de software que permite observar dados referentes a qualquer um dos computadores ligados em uma rede interna, ou seja, ele permite que sejam observados quais processos estão em execução e receber notificações de eventos que estão acontecendo em determinado computador. Os principais objetivos daquele trabalho foram:

a) coletar e visualizar informações de inventário de hardware de um computador conectado à rede;
b) visualizar os processos e serviços disponíveis e em execução no sistema operacional de um computador conectado à rede;

c) receber notificações do sistema operacional de um computador conectado a uma rede em eventos pré-definidos como, por exemplo, se o uso do seu processador ultrapassar 90%;
d) demonstrar o funcionamento do gerenciador de objetos através dos itens acima mencionados.

O protótipo desenvolvido neste trabalho visa complementar o trabalho de Braz (2003) disponibilizando para o administrador de rede informações de inventário de hardware e software de um computador conectado a rede, indicando automaticamente quando houver alguma alteração (instalação ou remoção de software, inclusão ou remoção de hardware).

5 DESENVOLVIMENTO DO TRABALHO

O objetivo do trabalho é desenvolver um protótipo de gerência de sistemas baseado no padrão WBEM utilizando o WMI. Neste capítulo serão abordados aspectos relevantes sobre o desenvolvimento do protótipo tais como os principais requisitos, a especificação e a implementação.

5.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Foram levantados alguns requisitos que devem estar presentes no protótipo. Estes requisitos demonstram algumas características que o protótipo precisa ter para que se alcance o resultado final desejado.

Os principais requisitos do protótipo são:

a) operar no sistema operacional Windows NT, 2000 ou XP;
b) coletar informações de classes do repositório de dados do WMI dos microcomputadores;
c) realizar um inventário de informações de um microcomputador em relação a instalação/remoção de softwares e hardwares, baseado em um comparativo com um histórico de inventário gerado anteriormente;
d) gerar relatórios de instalação/remoção de software e hardware e de inventário de microcomputadores;
e) enviar *e-mail* para o administrador da rede sobre toda alteração de inventário que ocorrer nos computadores da rede.

5.2 ESPECIFICAÇÃO

Para a especificação do protótipo foi utilizado a *Unified Modeling Language* (UML), utilizando o diagrama de casos de uso e o diagrama de atividades. Para auxiliar na construção destes diagramas foi utilizado a ferramenta *Rational Rose*.

5.2.1 DIAGRAMA DE CASOS DE USO

Na figura 10 está apresentado o diagrama de casos de uso do protótipo. Estes casos de uso identificam como o usuário interage com o sistema. Os principais casos de uso, descritos detalhadamente no diagrama de seqüência desta especificação são:

a) estabelecer conexão ao serviço WMI (**Conectar ao WMI**);

b) consultar informações para a realização de inventário(**Consultar Inventário**);

c) efetuar comparativo entre os inventários gerados (**Efetuar Comparativo**);

d) enviar um *e-mail* para o administrador da rede (**Enviar Email**).

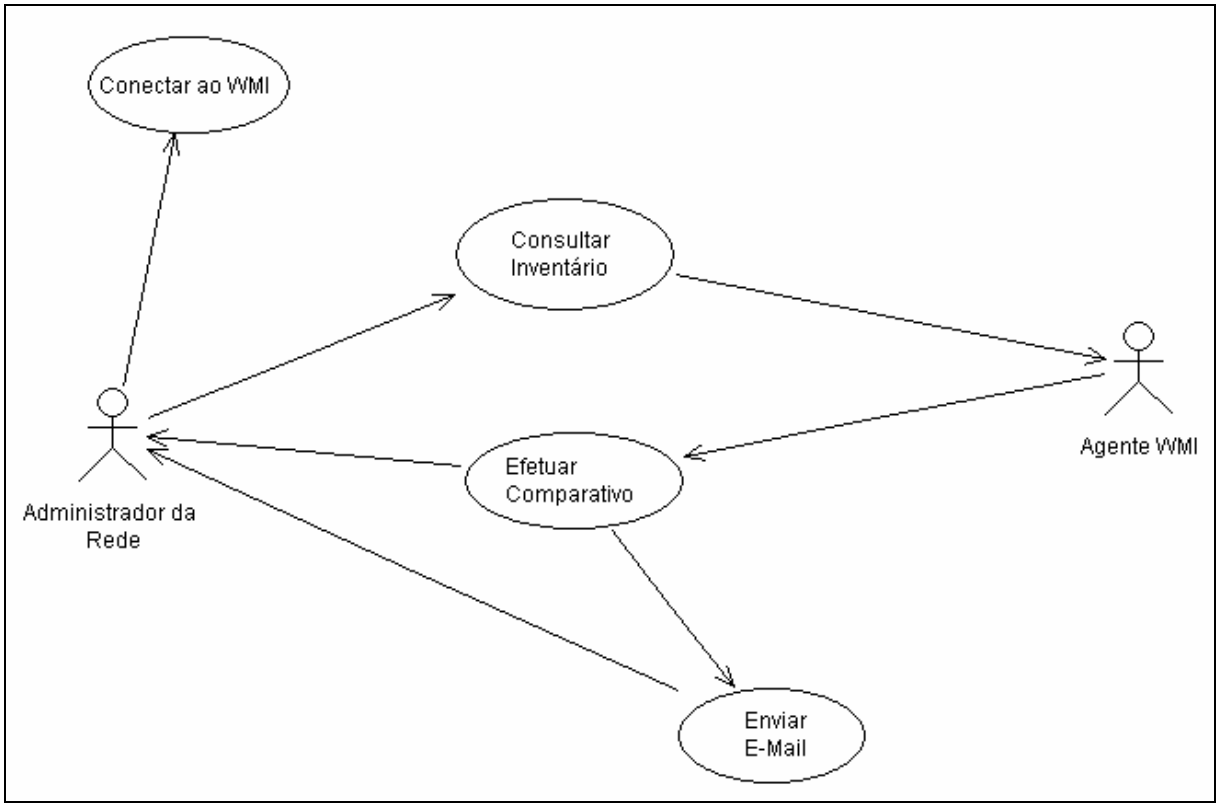


Figura 10 – Diagrama de Casos de Uso

5.2.2 DIAGRAMA DE ATIVIDADES

O diagrama de atividades descreve o processo em que o protótipo é executado, as ações executadas pelo administrador de rede e pelo agente WMI e a seqüência em que são executadas. O diagrama pode ser visto na figura 11.

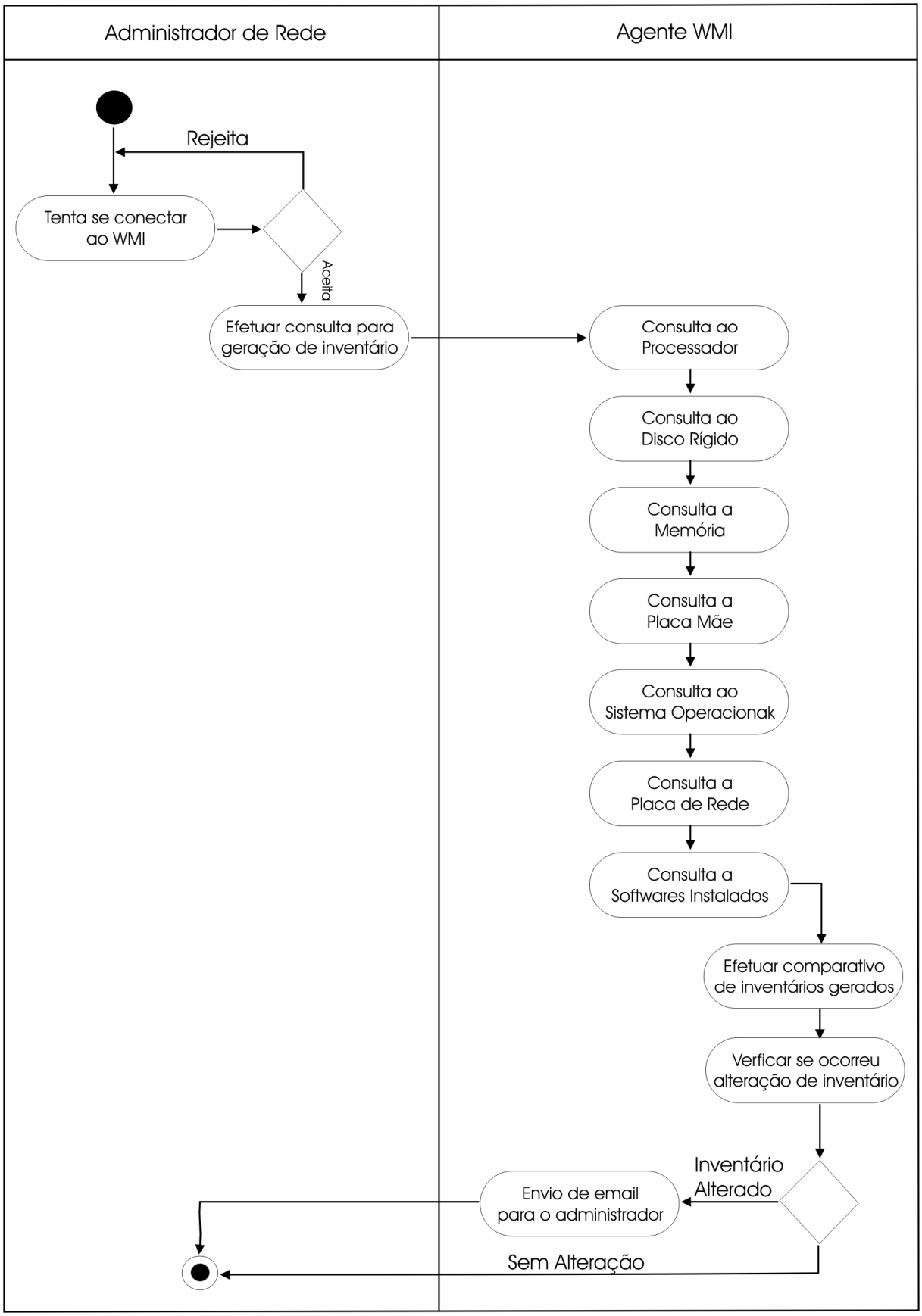


Figura 11 – Diagrama de atividades

5.3 IMPLEMENTAÇÃO

Nesta etapa é descrita a implementação do protótipo conforme a especificação descrita anteriormente.

5.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

O WMI, conforme mencionado anteriormente, utiliza em sua comunicação objetos distribuídos COM/DCOM, diferentemente da proposta do WBEM que é a utilização do protocolo HTTP. Esta comunicação foge do conceito fundamental que envolve a tecnologia de gerenciamento via WEB, onde as tarefas de gerenciamento são realizadas através de um *browser* (BRAZ 2003, p.44).

Neste trabalho foi abordado uma outra técnica. Foi utilizado o *namespaces System.Management*, que é um conjunto de classes implementadas pela Microsoft para interagir com o WMI, incluso na plataforma .NET utilizado para facilitar o desenvolvimento de aplicações de gerência sobre o WMI.

Para o desenvolvimento deste protótipo foram utilizados o ambiente de desenvolvimento *Borland Delphi* 8 .NET *Architect*, *framework* .NET, linguagem de programação *Object Pascal* e o sistema operacional Windows XP Professional.

O protótipo foi executado em uma rede interna, onde as estações gerenciadas estavam executando o serviço WMI e não possuíam nenhuma medida de segurança para evitar o acesso. É importante lembrar que o serviço WMI está disponível apenas nos sistemas operacionais Windows XP e Windows 2000. Nos sistemas mais antigos precisa ser instalado o *core* WMI, e além disto que o sistema suporte comunicação através de objetos distribuídos, sendo este configurado para permitir o acesso remoto.

O quadro 03 contém o código para a conexão de um computador com o WMI. Esse procedimento chama-se *ConectarWMI(host)*. Pode-se perceber que é necessário passar como parâmetro uma string ou seja o nome do computador ao qual deseja se conectar.

```
procedure conectaWMI(host : string);
var
  login : ConnectionOptions;
begin
  try
  begin
    login := ConnectionOptions.Create();
    frmprincipal.lbmicro.caption := 'Computador Conectado: ' + host;
    Scope :=  System.Management.ManagementScope.Create('\\' + host + '\root\CIMV2', login);
    Scope.Options.Impersonation := System.Management.ImpersonationLevel.Impersonate;
    Scope.Connect();
  end;
  except
  begin
    frmprincipal.lbmicro.caption := 'Computador Conectado: Nenhum';
    showmessage('Servidor não encontrado ou sem permissão de acesso!');
  end;
    exit;
  end;
end;
```

Quadro 03 – Conexão ao serviço WMI

No quadro 04 pode-se visualizar o código fonte da função *ConsultaClasse()* a qual faz consulta ao serviço WMI. Esta função é muito importante, pois todos os demais procedimentos irão executá-lo. Como parâmetro, a função irá receber uma *string* que representa a classe WMI a ser consultada e retornará uma lista com os objetos gerenciáveis pertencentes àquela classe.

```
function ConsultaClasse(Scope: ManagementScope;
  const WMIClassName: string): ManagementObjectCollection;
var
  Query: ObjectQuery;
  Searcher: ManagementObjectSearcher;
begin
  // Consulta ao WMI
  Query := ObjectQuery.Create('select * from ' + WMIClassName);
  Searcher := ManagementObjectSearcher.Create(Scope, Query);
  Result := Searcher.Get;
end;
```

Quadro 04 – Consulta de Classes

No quadro 05 pode-se visualizar o procedimento *WMISoftwareInfo(Scope)* implementado para visualizar a lista de softwares instalados no microcomputador através do *windows installer*. Pode-se ver a necessidade de passar parâmetros para a *ConsultaClasse()* onde é feito a consulta os serviço WMI. Os outros procedimentos seguem o mesmo funcionamento, apenas mudando o nome da classe a ser consultada.

```
//Informações referentes aos Softwares Instalados
procedure WMISoftwaresInfo(Scope: ManagementScope);
var
  QueryCollection: ManagementObjectCollection;
  Enumerator: ManagementObjectCollection.ManagementObjectEnumerator;
  Item: ManagementObject;
begin
  QueryCollection := ConsultaClasse(Scope, 'Win32_Product'); // ou CIM_Product
  Enumerator := QueryCollection.GetEnumerator;

  frmprincipal.dados.lines.add('');
  frmprincipal.dados.lines.add('Informacao sobre Programas Instalados');
  frmprincipal.dados.lines.add('-------------------------------------');
  while Enumerator.MoveNext do
  begin
    Item := Enumerator.Current as ManagementObject;
    with Item, Console do
    begin
      frmprincipal.dados.lines.add('     Programa  : ' + GetPropertyValue('Caption').ToString);
    end;
  end;
  frmprincipal.dados.lines.add('-----');
end;
```

Quadro 05 – Procedimento para consulta de informação de Instalação de Softwares

No quadro 06 pode-se visualizar a implementação do procedimento para o envio de *e-mail* para o administrador da rede, alertando sobre alguma eventual alteração em algum dos microcomputadores da rede interna. O procedimento é chamado após a geração de novo inventário de computador e após a comparação dos dados existentes. O procedimento verifica se houve alguma alteração de inventário do computador consultado. Caso tenha ocorrido alteração de inventário, é enviado um *e-mail* com os dados e o nome do computador que teve alteração de inventário para o administrador da rede.

```
procedure envioemail;
var
  Mensagem : System.Web.Mail.MailMessage;
  ArqErro : TextFile;
  LinhaErro : String;
  LinhasMensagem : String;
  Arq : TextFile;
  linha : string;
  config : array[0..2] of string;
  i : integer;
  dir : string;
begin
  assignfile(ArqErro, dir + 'ArquivoLog\ERRO.TXT');
  reset(ArqErro);
  readln(ArqErro, LinhaErro);
  dir := ExtractFilePath(Application.ExeName);
  assignfile(Arq, dir + '\Config.TXT');
  reset(Arq);
  i := 0;
  while not eof(Arq) do
  begin
    readln(Arq, linha);
    config[i] := linha;
    i := i + 1;
  end;
  frmenvio.show;
  Mensagem :=  System.Web.Mail.MailMessage.Create();
  Mensagem.From := config[2];// 'rodrigo@brwnet.com.br';
  Mensagem.Set_To(config[1]); //'msbinformatica@metuno.com.br'
  frmenvio.lbpara.Caption := 'Para: ' + config[1];//'msbinformatica@metuno.com.br';
  Mensagem.Subject := LinhaErro;
  frmenvio.lbassunto.caption := 'Assunto: ' + LinhaErro;
  frmenvio.Repaint;
  //incluir dados no corpo da mensagem de texto
  LinhasMensagem := '';
  While not eof(ArqErro) do
  begin
    readln(ArqErro, linhaErro);
    LinhasMensagem := LinhasMensagem + linhaErro + #13;
  end;
  closefile(ArqErro);
  frmenvio.Repaint;
  Mensagem.Body := LinhasMensagem;
  System.Web.Mail.SmtpMail.SmtpServer := config[0]; //'smtp.indusul.com.br';
  frmenvio.Repaint;
  if LinhasMensagem <> '' then
  begin
    System.Web.Mail.SmtpMail.Send(Mensagem);
    frmenvio.lbprogesso.caption := 'Email enviado com sucesso!';
    frmenvio.close;
    Mensagem.Free;
  end
  else
    Mensagem.Free;
end;
```

Quadro 06 – Envio de e-mail para o administrador

5.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Conforme descrito anteriormente, para a utilização do protótipo é necessário:

a) no servidor, onde será executado o protótipo, deve estar instalado o *framework* .NET;

b) nas estações, não há necessidade de nenhuma ferramenta instalada, apenas o WMI deve estar rodando.

A seguir pode-se visualizar as telas do protótipo, bem como sua descrição em relação a interação com o usuário. Para utilizar o protótipo é necessário apenas executar o programa.

Na figura 12 pode-se visualizar a tela principal do protótipo que existem três botões, que é utilizado para o usuário poder interagir com o sistema: Geração de Inventário, Comparativo, Lista Computadores. Quando o sistema é executado, ele não se conecta com nenhum dos computadores da rede, e por isso o usuário terá que pressionar o botão Lista Computadores primeiro para poder selecionar a qual computador deseja se conectar. Após conectado, o usuário deverá pressionar o botão Geração de Inventário para gerar o inventário do computador selecionado. E por fim pressionar o botão Comparativo para efetuar o comparativo entre o arquivo gerado e o modelo de inventário que já foi salvo anteriormente. Na primeira vez que for feito o comparativo de arquivos de inventário ele não executa o comparativo, pois primeiramente ele cria o arquivo de inventário que servirá como base para executar comparativos posteriores.

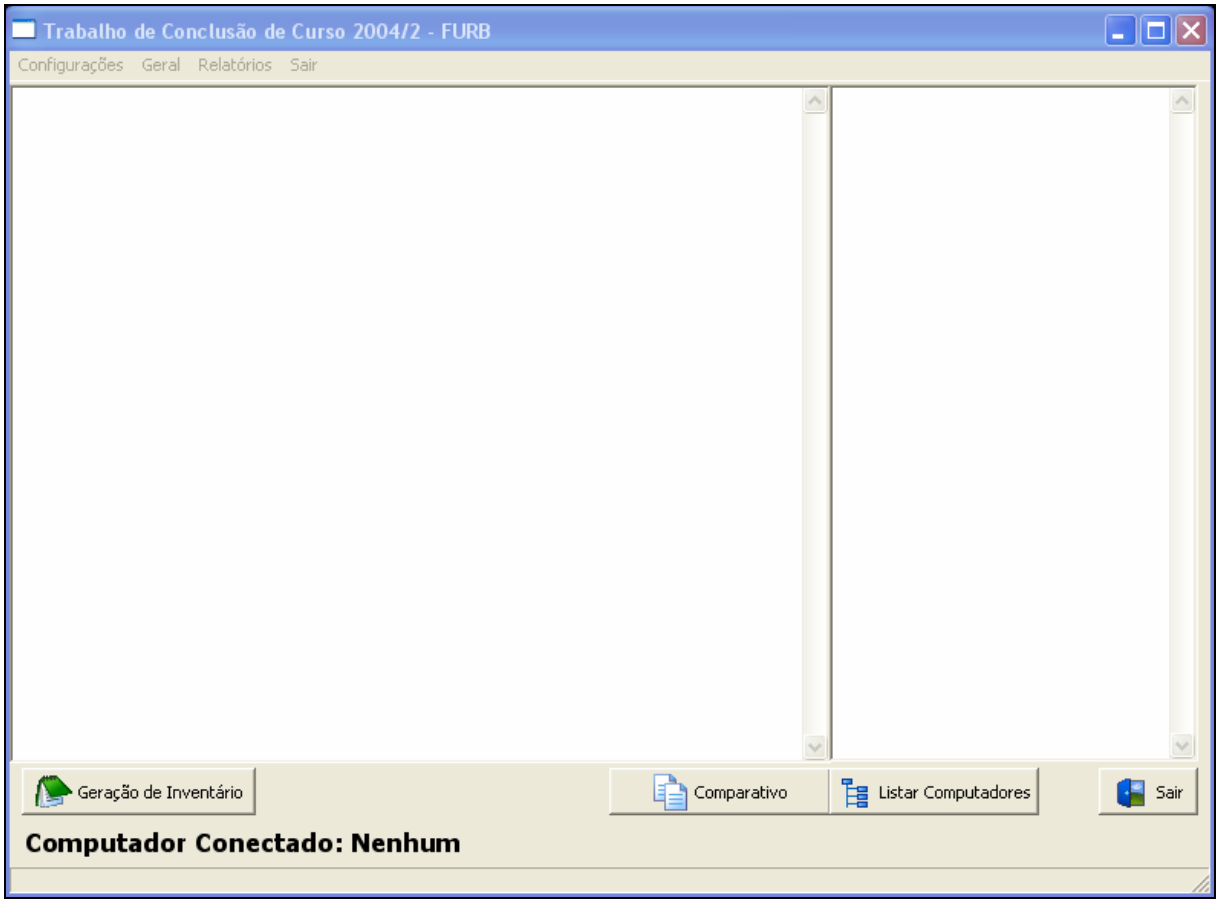


Figura 12 – Tela principal do protótipo

Na figura 13 pode-se visualizar o protótipo em execução, observado que no menu Geral é permitido que seja gerado o inventário de apenas alguns dados, ou seja não ha necessidade de gerar o inventário completo, sendo possível escolher entre todos as classes ou selecionar as que desejar. A possibilidade de escolher apenas alguma das classes não permite que seja feito um comparativo correto, pois o modelo de inventário utilizado para o comparativo a geração de um inventário completo, e se for feito uma geração parcial, irá ocorrer diversos erros no comparativo.

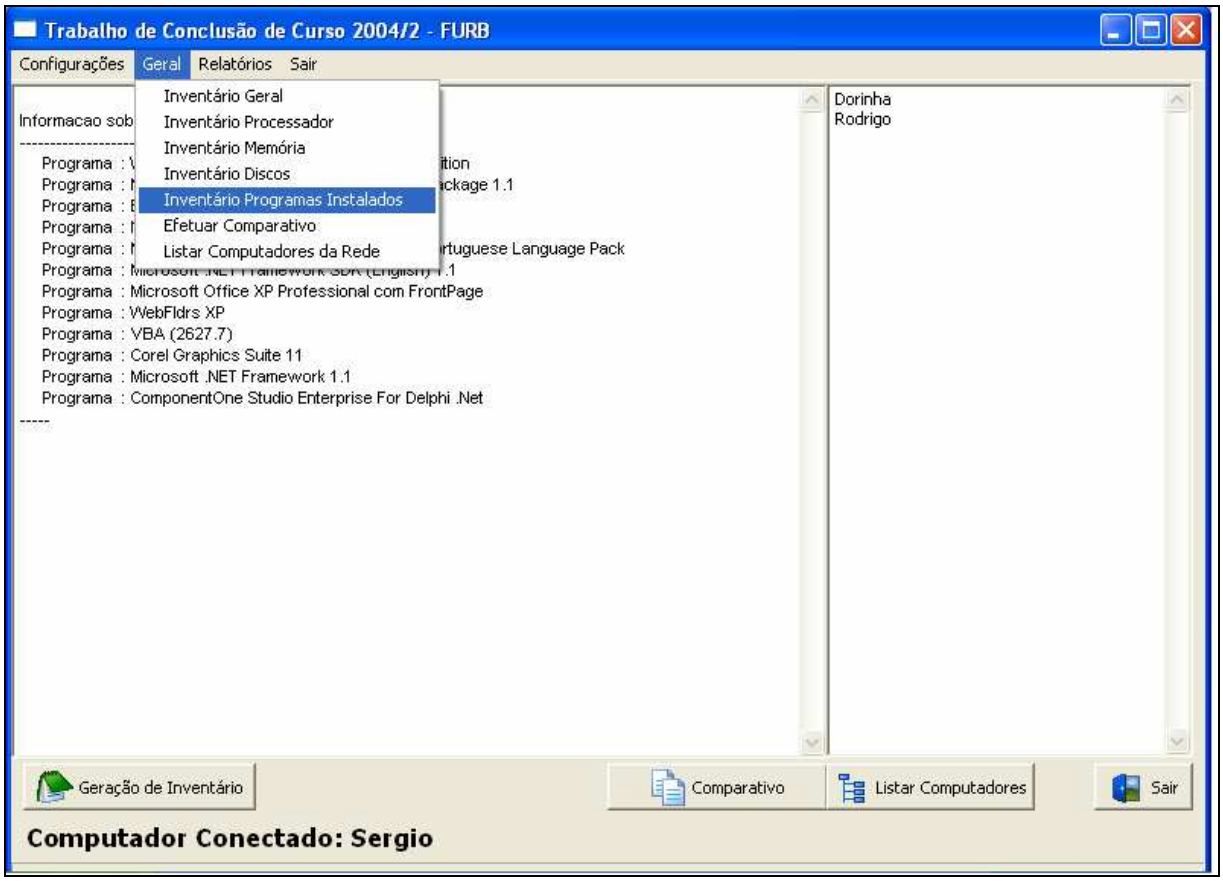


Figura 13 – Tela de consulta de classes do WMI

Na figura 14 pode-se visualizar o envio de *e-mail* para o administrador da rede, com as devidas alterações efetuadas em algum dos computadores da rede. Nesta tela pode ser visto também a configuração utilizada para o envio do *e-mail* como servidor SMTP e o destinatário do *e-mail*. Estes dados podem ser alterados no menu de configurações, pois são armazenados no diretório onde o protótipo está instalado, em um arquivo de texto, o “Config.txt”.

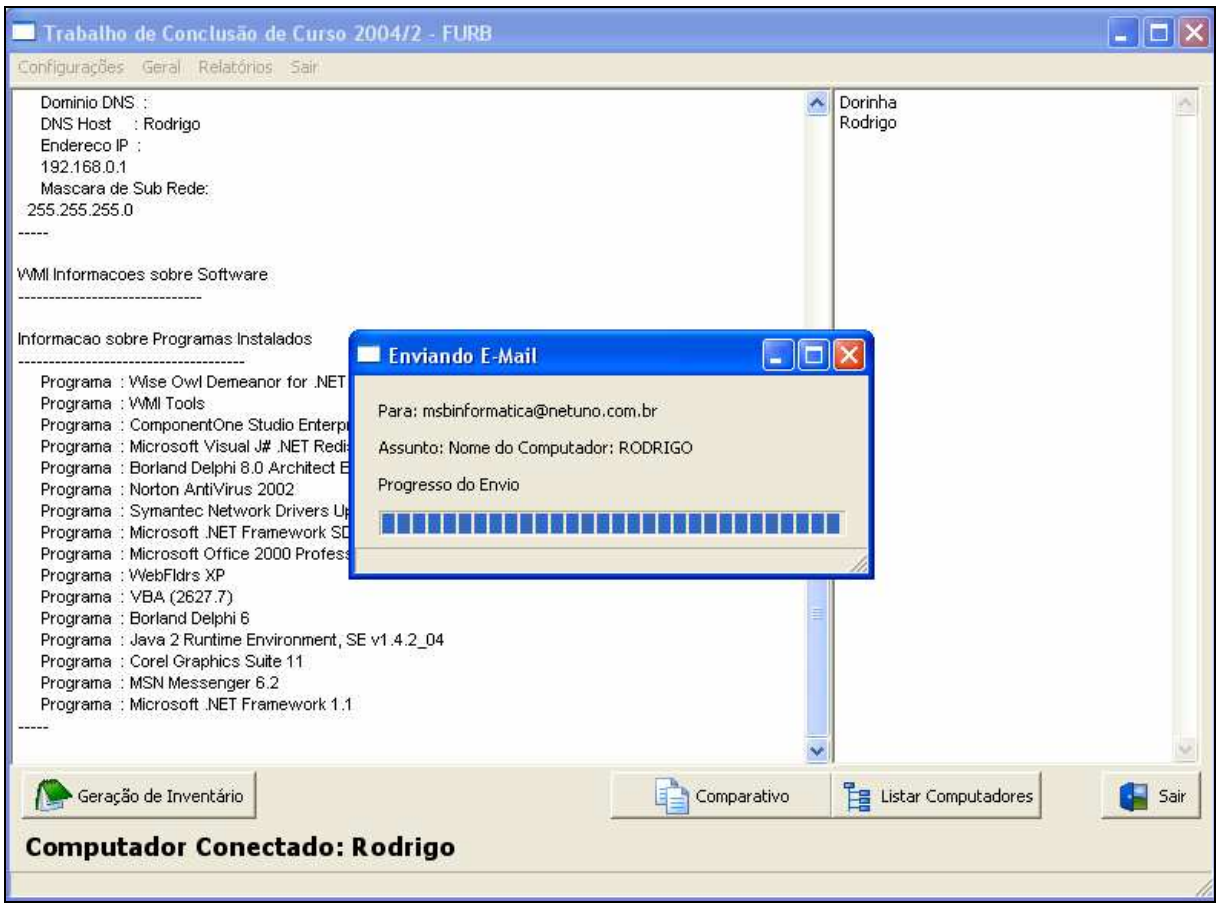


Figura 14 – Envio de E-Mail

Na figura 15 pode-se visualizar a busca do nome dos computadores da rede. Esta busca foi implementada utilizando o Delphi 6 (por não ter conhecimento de algum tipo de consulta do nome dos computadores conectados a rede no Delphi 8), e sendo feito dentro do Delphi 8 uma chamada de um programa externo. Quando este programa é chamado ele cria um arquivo de texto com o nome dos computadores conectados na rede e o protótipo desenvolvido apenas lê os dados no arquivo texto e escreve na tela.

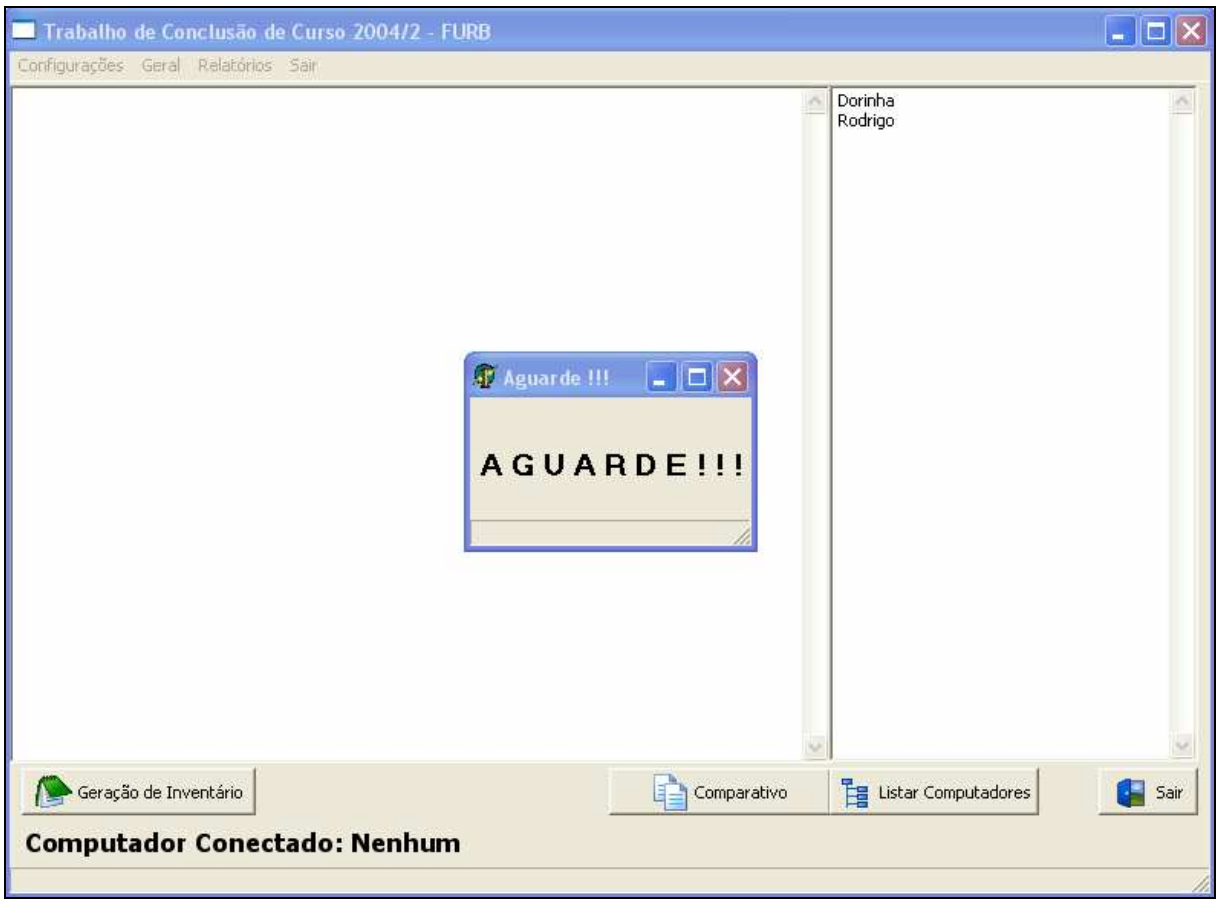


Figura 15 – Consulta de computadores da rede

5.4 RESULTADOS E DISCUSSÃO

Os módulos de consulta de processador, placa mãe, discos, placa de rede e configurações de rede atenderam os requisitos do protótipo, sendo possível a geração de inventário de dados do computador consultado, porém somente poderá ser feito a consulta onde o protótipo estiver instalado, pois o mesmo não foi desenvolvido para ser utilizado via internet.

O módulo de consulta de softwares não atendeu os requisitos, pois não foi possível consultar todos os softwares, apenas os que utilizam o *windows installer* para serem instalados, e não todos os softwares instalados e registrados no registro do Windows.

Até o momento não foi possível aperfeiçoar este módulo. Foram realizadas diversas consultas a WEB para encontrar uma solução diferente, mas ficou constatado que não é possível a consulta de todos os softwares instalados, devido a limitação do serviço WMI.

6 CONCLUSÕES

Com o desenvolvimento deste trabalho pode-se concluir que a utilização de tecnologias baseadas na arquitetura WEB é uma tendência. Cada vez mais se observa o uso de técnicas e ferramentas voltadas para a WEB no desenvolvimento de aplicações. Isto se tornou uma realidade na área de gerência devido ao surgimento do padrão WMI.

O padrão WMI não representa apenas um modelo em que as informações são exibidas num *browser*, mas também uma forma de reduzir custos e facilitar o gerenciamento, sendo que o WMI, implementação proposta pela Microsoft, mostrou-se adequado para o gerenciamento de estações que utilizam sistemas operacionais da Microsoft.

Com a utilização do WMI é possível realizar operações de gerenciamento que vão desde a criação de um inventário até instalações de software remotamente.

O protótipo desenvolvido limita-se apenas a mostrar os dados de inventário, e não possui nenhuma interação destes dados com o usuário, como por exemplo, alteração de dados referentes à placa de rede.

A ferramenta de gerenciamento auxilia o administrador da rede a monitorar a instalação e remoção de softwares e hardwares, bem como visualiza informações de inventário de um computador conectado a rede. Assim efetua o comparativo entre arquivos de inventário, verificando alguma alteração ocorrida. Desta forma envia o e-mail ao administrador da rede notificando-o de alguma alteração ocorrida nos computadores.

Uma restrição a esta proposta deve-se ao fato de que é possível gerenciar apenas estações que estão rodando o serviço WMI. Este serviço é disponível apenas em computadores que rodam o Windows. O protótipo também exige que esteja instalado o *framework* .NET, e este apenas está disponível para o sistema operacional da Microsoft.

Como contribuições do trabalho pode-se citar:

a) um estudo sobre os conceitos de gerência corporativa;
b) um estudo sobre a plataforma de desenvolvimento .NET;
c) um estudo sobre o padrão de gerência WMI;
d) um estudo sobre a plataforma de desenvolvimento Borland Delphi 8 .NET;

e) a utilização do *namespaces System.Management* e da plataforma .NET no desenvolvimento de aplicações de gerenciamento de sistemas utilizando o WMI.

6.1 EXTENSÕES

Como sugestão para trabalhos futuros pode-se citar:

a) implementação de um módulo que permita a consulta de todos os softwares instalados;
b) implementação de um módulo utilizando o .NET para busca dos nomes dos computadores da rede em que estiver conectado;
c) implementar um módulo em que possa ser observado a tela do computador a ser consultado.

REFERÊNCIAS BIBLIOGRÁFICAS

BRAZ JUNIOR, Edson Luis da Silva. **Protótipo de um software para gerência de sistemas baseado no padrão WBEM utilizando o WMI**. 2003. 63 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CARVILHE, José Luís Vieira. **A utilização de tecnologias WEB em sistemas de gerência corporativa**, Curitiba, 2000a. Disponível em: <http://www.pr.gov.br/celepar/batebyte/edicoes/2000/bb99/gerencia.htm>. Acesso em: 07 ago. 2004.

CARVILHE, José Luís Vieira. **Web Based Enterprise Management – WBEM**, Curitiba, 2000b. Disponível em: <http://www.pr.gov.br/celepar/celepar/batebyte/edicoes/2000/bb101/web.htm>. Acesso em: 07 ago. 2004.

D'ANGELO, Fernando. **Microsoft .NET. A plataforma Java da Microsoft**, 2003. Disponível em: <http://www.aspbrasil.com.br/conteudo/detalhesCompleta.aspx?codConteudo=3306&Secao=TUTORIAIS>. Acesso em: 07 ago. 2004.

DMTF - DISTRIBUTED MANAGEMENT TASK FORCE. **CIM Tutorial**. 2002. Disponível em: <http://www.wbemsolutions.com/tutorials/CIM/metaschema.html>. Acesso em: 08 ago. 2004.

DMTF - DISTRIBUTED MANAGEMENT TASK FORCE. **Common information model (CIM) specification version 2.2**, [s.1], 1999. Disponível em: <http://www.dmft.org/standards/cim_spec_v22/>. Acesso em: 21 mar. 2004.

IMASTER. **A .NET FRAMEWORK.** 2003. Disponível em: < http://www.imasters.com.br/artigo.php?cn=1032&cc=76>. Acesso em: 08 ago. 2004.

MARTINSSON, Tobias. **Desenvolvendo scripts XML e WMI para o Microsoft SQL Server 2000**. São Paulo: Makron Books, 2002.

MICROSOFT CORPORATION. **Learn- WMI**, [s.1], 1999. Disponível em: <http://www.microsoft.com/downloads/release.asp?releaseid=12570>. Acesso em: 21 mar. 2004.

MICROSOFT. **Best Practices for Delegating Active Directory Administration** [s.1], 2000. Disponível em: <http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/directory/activedirectory/actdid1.mspx>. Acesso em: 08 ago. 2004.

MICROSOFT. **WMI – Windows Management Instrumentation**. [s.1], 2002. Disponível em: <http://www.microsoft.com/hwdev/driver/WMI/default.asp>. Acesso em: 08 ago. 2004.

MOORE B., ELLESSON E., STASSNER J, WESTERINEN, A. **Policy Cire Information Model – Version 1 Specification** 2001.Disponível em: <http://www.ietf.org/rfc/rfc3060.txt?number3060>. Acesso em: 07 ago. 2004.

SPECIALSKI, Elizabeth Sueli. **Gerência de redes de computadores e de telecomunicações**, Florianópolis, [2002]. Disponível em: < http://notes.ufsc.br/aplic/beth.nsf/ccae10dff955ca3f0425694b0078fbce?OpenForm>. Acesso em: 07 ago. 2004.