

# ProGerWMI: Utilizando WMI para a Construção do Protótipo de uma Ferramenta para o Gerenciamento de Redes

Jorges Borges Figueiredo, Edeilson Milhomem da Silva, Fabiano Fagundes

Sistemas de Informação – Centro Universitário Luterano de Palmas (CEULP/ULBRA)  
Av. Teotônio Segurado, 1501 Sul Cx. Postal 160 - CEP 77054-970 Palmas-TO

{jorge,milhomem,fagundes}@ulbra-to.br

**Resumo.** *Este trabalho objetiva apresentar o desenvolvimento do protótipo de uma ferramenta para a gerência de redes Windows, utilizando tecnologias presentes no Windows nativo, como Windows Management Instrumentation (WMI), HTML Application (HTA) e VBScript, a fim de manter a ausência de pré-requisitos para a sua instalação/execução, isto é, a não necessidade de se instalar qualquer software ou componente para que a ferramenta possa ser utilizada.*

## 1. Introdução

Atualmente, as redes de computadores, devido às facilidades que proporcionam, tornam-se cada vez mais utilizadas pelas empresas. As suas aplicações afetam não somente as áreas funcionais das corporações mas, também, seus parceiros, fornecedores e até mesmo os próprios clientes. A sua importância está diretamente relacionada ao tamanho, complexidade e heterogeneidade no que diz respeito à variedade de tecnologia e fornecedores nos dispositivos de uma mesma rede.

Em função da dimensão tomada pelas redes de computadores, torna-se cada vez mais crucial a manutenção dos seus serviços aos usuários. Para garantir, ou pelo menos, diminuir ao máximo as falhas e suspensão desses serviços, faz-se necessária a utilização da gerência de redes que busca, por meio da utilização de ferramentas, manter um controle sobre o que está acontecendo em uma determinada rede.

O processo de gerenciamento, segundo Specialski (2002), pode ser encarado como a monitoração, o controle e a configuração dos diversos dispositivos presentes em uma rede (computadores, *switches*, *routes*, etc). É neste ponto que entram as ferramentas de gerenciamento, que procuram automatizar, se não todas, mas pelo menos uma grande parte das tarefas de gerência. Para isso, essas ferramentas trabalham o monitoramento e a detecção de problemas, além de possibilitar que muitas correções sejam feitas remotamente por meio de configurações nos dispositivos que compõem a rede. Um dos problemas encontrados na utilização destas ferramentas é que os administradores esperam que elas atendam todas as suas necessidades de forma simples e automática.

Isso ocorre porque, ou são esperados verdadeiros “milagres”, ou seus componentes não são utilizados de forma correta, sub-utilizando os recursos que a ferramenta oferece. Essa sub-utilização pode se dar pelo fato de algumas ferramentas possuírem muitos recursos aglomerados. Essa quantidade excessiva de recursos, na

grande maioria das vezes, pode confundir os administradores e, conseqüentemente, muitos recursos não são utilizados adequadamente.

Buscando minimizar esses problemas, este trabalho apresenta o Protótipo de uma Ferramenta para Gerência de Redes *Windows* (ProGerWMI), que dá aos administradores funcionalidades simples, porém úteis no dia-a-dia do gerenciamento de redes, como, por exemplo, gerenciar processos remotamente, efetuar *logoff* de usuários, gerenciar serviços, etc. Outro ponto positivo do ProGerWMI é demonstrar a facilidade de se desenvolver suas próprias ferramentas que atendam exatamente as suas necessidades e, tudo isso sem nenhum custo adicional, além do próprio sistema operacional. Para atingir tais objetivos, o ProGerWMI foi desenvolvido utilizando recursos de gerência disponíveis no *Microsoft Windows* nativo, como WMI, HTA e a linguagem *VBScript*, buscando-se ao máximo eliminar a necessidade de pré-requisitos de instalação e execução.

## 2. WMI

O WMI é uma tecnologia que permite a manipulação de informações para o gerenciamento e controle de redes corporativas. Essa tecnologia é distribuída como um componente nativo dos sistemas operacionais *Windows 2000* e *XP*, e como um pacote para *download* para os sistemas mais antigos MSDN (2005e).

Essa instrumentação permite que administradores consultem, monitorem e até configurem, sistemas operacionais, aplicações de rede, dispositivos físicos, etc. Sendo que essas tarefas podem ser realizadas de uma forma bastante unificada, transparente e padronizada.

O WMI é baseado na iniciativa WBEM (*Web-Based Enterprise Management*), desenvolvida pela DTMF (*Distributed/Desktop Management Task Force*) DMTF (2005). O WBEM é uma iniciativa da indústria, que busca o desenvolvimento de tecnologias padrões para o acesso às informações de gerenciamento em ambientes corporativos MSDN (2005c). Esse padrão foi inicialmente proposto por um grupo de empresas de tecnologia dentre as quais estava *Microsoft*. Atualmente as responsabilidades pela manutenção e desenvolvimento do padrão são da DMTF.

Logo, o WMI é uma implementação do WBEM que inclui objetos gerenciados e definidos por esse modelo, bem como extensões a ele, a fim de dar suporte a informações adicionais disponíveis na plataforma *Windows* MSDN (1999).

Essa tecnologia permite que o gerenciamento seja realizado através de uma interface simples, orientada a objetos e extensível. Outro ponto positivo é que qualquer aplicação WMI pode acessar máquinas locais ou remotas de forma transparente e através da mesma interface. A seguir são apresentadas algumas características do WMI mais fundamentais na resolução de problemas de gerenciamento enfrentados por administradores de redes corporativas MSDN (1999):

- **API de *Script* Uniforme:** todos os objetos gerenciados (componentes passíveis de gerenciamento como: dispositivos físicos, aplicações, etc) são modelados sob

uma estrutura comum de objetos baseada no modelo de objetos CIM do padrão WBEM. Dessa forma, o acesso a diferentes fontes de informação é feito através da utilização de uma única API, ou seja, a API WMI;

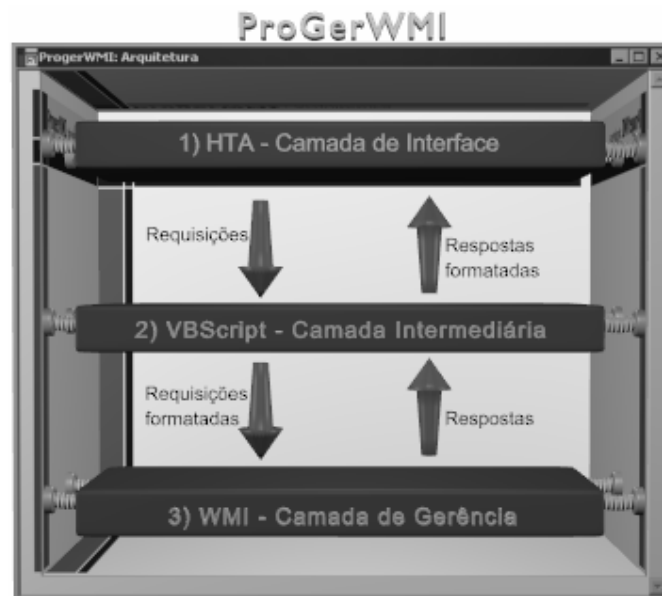
- **Administração Remota:** dentro do WMI as definições dos objetos gerenciados estão disponíveis para aplicações e *scripts*, tanto localmente quanto remotamente. Dessa maneira é possível gerenciar objetos remotos sem trabalho adicional;
- **Navegação e Pesquisa:** aplicações e *scripts* são aptos a descobrir quais informações são disponíveis sobre o sistema pela enumeração das classes existentes. A relação entre objetos próximos pode ser detectada e cruzada para se saber como uma entidade gerenciada afeta a outra. É possível, por exemplo, se gerenciar as portas as quais um *modem* está conectado pela busca das associações existentes entre a classe que representa o *modem* e a que representa as portas;
- **Capacidades de Consulta:** O WMI gerencia seus dados de forma similar a uma base de dados relacional. Em função disso, ele permite a realização de consultas através de uma extensão da linguagem SQL, possibilitando o enfoque das requisições somente nos dados de interesse;
- **Publicação e Subscrição de Eventos:** eventos podem ser acionados por qualquer mudança em qualquer objeto gerenciado do sistema, independente se o mesmo suporta internamente a capacidade de eventos ou não. Assinantes de eventos são aptos a receberem notificações de cada evento específico, baseados no seu interesse particular, que de preferência só deve possibilitar que o mesmo pegue eventos que tenham sido pré-definidos pelos seus desenvolvedores originais. Na prática, a arquitetura é bem flexível e permite que qualquer ação definida pelo usuário possa ser interpretada como um evento.

Das características acima, somente a última não foi utilizada no desenvolvimento ProGerWMI, pois a utilização de eventos é mais interessante para a obtenção de relatórios que descrevem como se comportou um determinado recurso gerenciado durante um período de tempo, ou para se executar ações quando da ocorrência de um dado evento. Como essas tarefas exigiriam o desenvolvimento de uma forma de apresentação mais elaborada, não foram implementadas nesta primeira versão do ProGerWMI.

### 3. Arquitetura do ProGerWMI

Para um melhor entendimento do protótipo foi desenvolvida uma arquitetura geral (Figura F1), organizada no formato de camadas. A camada inicial é a de interface ou apresentação, que tem a responsabilidade de coletar comandos passados pelo administrador e de exibir os resultados quando os dados são retornados pela segunda camada. A segunda, a camada intermediária, é responsável pelo tratamento dos dados (formatação e validação) e transmissão desses mesmos dados à camada WMI. Essa

camada é responsável, também, pelo tratamento dos dados retornados pela camada WMI, que, posteriormente, são repassados à camada de apresentação. Por fim, tem-se a camada WMI que é a que efetivamente realiza as operações de gerência, ou seja, tanto lê informações dos recursos gerenciáveis quanto os configura, com base nas solicitações feitas pela camada intermediária.



**Figura F1: Arquitetura do ProGerWMI (visão interna)**

Conforme a Figura F1, que representa a arquitetura do ProGerWMI, há a presença de três camadas: camada 1 – essa camada é constituída pela linguagem HTML rodando no ambiente HTA; camada 2 – esta é formada pela linguagem de script *VBScript* e é responsável pelo transporte dos dados entre a camada 1 e a camada 2; e, por fim, camada 3 – que é constituída pela API WMI. As seções seguintes fazem uma abordagem detalhada a respeito das referidas camadas.

### 3.1 Camada de Interface

O componente principal dessa camada é a linguagem HTML (*HyperText Markup Language*) W3C (2005), que é utilizada para criar recursos de interface para o usuário. Essa linguagem é comumente utilizada para a criação de página WEB, que são exibidas por navegadores como o *Internet Explorer*, por exemplo. Entretanto, no desenvolvimento do ProGerWMI, o ambiente responsável por essa exibição é o *HTML Application*, um ambiente criado especialmente para a execução de aplicações.

HTAs são aplicações completamente autônomas. Elas são confiáveis e mostram somente os menus, ícones, barras de ferramentas, e informações nos títulos que o desenvolvedor cria. Em resumo, HTAs mantêm as capacidades do *Internet Explorer*, como o seu modelo de objetos, performance e características de renderização, porém sem forçar a utilização do modelo de segurança rígido e interface de usuário do navegador MSDN (2005a).

A escolha da HTML deu-se pelo fato da mesma possibilitar a criação de recursos gráficos que facilitam consideravelmente a interação do usuário com a ferramenta e também pela possibilidade de integração entre a HTML e a linguagem utilizada na camada intermediária (VBScript). É importante ressaltar que a utilização da HTML foi viabilizada pela utilização do HTA (*HTML Application*). O uso do HTA fez-se necessário porque no contexto de segurança de um navegador *web* não seria possível acessar a API WMI de forma transparente ao usuário. Essa transparência é fundamental, pois evita que alertas de segurança sejam enviados ao usuário durante o acesso aos objetos de gerenciamento WMI. Outra vantagem do HTA, já citada anteriormente, é o fato dele possibilitar, também, que a interface da ferramenta não seja vinculada ao navegador. A Listagem L1 apresenta um exemplo da utilização dos recursos do HTA.

```
01 <hta:application
02 id="criarcomp"
03 applicationname="criarcomp"
04 icon="ProgerWMI.ico">
```

#### Listagem L1: Alguns atributos exclusivos dos HTAs

A Listagem L1 mostra algumas das possibilidades de configurações adicionais ao HTML comum, que o HTA permite. Através da utilização da marcação (linha 01) é possível ter acesso ao modelo de objetos estendido das aplicações HTA. Na linha 02 é definido o atributo `id`. É através deste valor que os dados da linha de comando, por exemplo, são acessados. Na linha 03 é definido o atributo `applicationname`, que identifica a aplicação, independentemente de qual arquivo físico está sendo usado para executá-la. Esse atributo é particularmente útil quando o atributo `singleinstance` é definido como verdadeiro; é através do `applicationname` que o HTA verifica se a instância é única quando o `singleinstance` é definido. Por fim, tem-se o atributo `icon`, que recebe como valor o caminho físico do arquivo de imagem que será utilizado como ícone da aplicação (linha 04). A Listagem L2 exhibe o trecho de código que acessa a linha de comando do HTA:

```
01 collCmdLn = split(criarcomp.commandLine, "*")
02 if UBound(collCmdLn)>0 then
03   user = collCmdLn(UBound(collCmdLn)-1)
04   passWd = collCmdLn(UBound(collCmdLn))
05 End if
```

#### Listagem L2: Acesso às propriedades dos HTAs

Conforme representação da Listagem L2, na linha 01 a propriedade `commandLine` é acessada através do valor definido para o atributo `id` na linha 02 da Listagem L1. Todos os atributos definidos na marcação `<hta:application>`, que possuem as respectivas propriedades, podem ser acessados programaticamente através do valor definido no atributo `id`.

### 3.2 Camada Intermediária

A base dessa camada é a linguagem *Visual Basic Script (VBScript)*, que é derivada da linguagem *Visual Basic* que possibilita a utilização de *scripts* em uma variedade de ambientes como clientes *web* no *Internet Explorer* e servidores *web* no *Internet Information Server* MSDN (2005b). Outro ambiente que suporta essa linguagem é o HTA; em função disso optou-se pela escolha da *VBScript* para a realização da implementação nessa camada.

O *VBScript* comunica-se com a aplicação hospedeira, no caso o HTA, através do *Windows Script*, que é um componente do *Windows* responsável por encapsular a utilização de *scripts* de forma unificada em diferentes aplicações hospedeiras. Dessa forma, não há a necessidade da utilização de códigos especiais para realizar a integração entre o *VBScript* e os componentes oferecidos pela aplicação hospedeira MSDN (2005b). Por essas características a linguagem *VBScript* mostrou-se adequada para a realização da integração entre as camadas de interface e a WMI.

Na teoria, seria possível realizar as tarefas de gerência somente com essa camada e a seguinte, entretanto, na prática, a forma de interação seria muito complicada com a ausência de uma interface gráfica, o que inviabilizaria a usabilidade do ProGerWMI. A Listagem L3 apresenta um trecho de código que demonstra o papel da linguagem *VBScript* na efetivação da comunicação entre as duas camadas das extremidades. A função escolhida para exemplificar essa comunicação foi a de criação e finalização de processos.

```
01 sub Executar()  
02 if document.all.findar.checked then  
03     prcss = document.all.txtNome.value  
04     if IsNumeric(prcss) then  
05         strQuery = " where processId=" & prcss  
06     else  
07         if StrComp(right(prcss,4), ".exe",1) = 0 then  
08             prcss = left(prcss, len(prcss)-4)  
09         end if  
10         strQuery = " where name=" & prcss & ".exe" & ""  
11     end if  
12 end if  
13 set collPrcss = conecta().ExecQuery("Select * from  
Win32_Process" & strQuery)  
14 if IsEmpty(collPrcss) then  
15     ret.children(1).innerText = ""  
16     document.all.msg.innerText = "A conexão falhou!"  
17     exit sub  
18 end if  
19 if document.all.findar.checked then  
20     for each objPro in collPrcss  
21         if objPro.Terminate()=0 then  
22             document.all.msg.innerText = "Operação realizada com  
sucesso."  
23             exit sub  
24         end if  
25     next  
26     document.all.msg.innerText = "Operação falhou!"  
27     exit sub  
28 end if
```

FIGUEIREDO, Jorge Borges; SILVA, Edeilson Milhomem; FAGUNDES, Fabiano. ProGerWMI: Utilizando WMI para a Construção do Protótipo de uma Ferramenta para o Gerenciamento de Redes. In: VII ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO ESTADO DO TOCANTINS, 2005, Palmas. **Anais...** Palmas: 2005.

```
29     cont = 0
    ...
```

### **Listagem L3: Utilização da linguagem VBScript (Criação e Finalização de processos)**

A primeira verificação feita nesse código é se a operação escolhida é a de finalização (linha 02). Caso seja, é necessário montar a cláusula que retornará, da coleção de processos em execução na máquina, apenas o que tiver o nome ou PID igual ao fornecido pelo usuário através da camada de interface. Para tanto, é necessário saber se este valor é um número ou não. Se for número, o mesmo deverá ser comparado com o PID, no caso, a propriedade `processId` do objeto `Win32_Process` (linhas 04 e 05). E, se for o nome, deverá ser comparado à propriedade `name` do referido objeto (linhas 07 a 10). Na linha 13 é feita a chamada à função de conexão, utilizando o `ExecQuery`, que retorna uma coleção de objetos instanciados, levando em conta os critérios da consulta passada como parâmetro.

É possível notar que os dados colhidos pela camada de interface foram capturados pela camada intermediária, que os formatou e os encaminhou para os objetos e métodos adequados na camada WMI. Da mesma forma o caminho inverso é feito, ou seja, os dados retornados pela camada WMI, a partir das solicitações da camada intermediária serão formatados e repassados à camada de interface para a exibição ao usuário.

### **3.3 Camada de Gerência**

O principal componente dessa camada é também a base do ProGerWMI, isto é, a API WMI. Essa API fornece acesso aos objetos gerenciáveis da rede. Acesso esse, feito através do WMI, que é uma implementação da *Microsoft* para o padrão WBEM, como explicado na seção 2.

Nessa camada é feita a manipulação dos objetos gerenciáveis diretamente. Ela é compreendida por toda a API WMI, que é uma abstração dos recursos gerenciáveis feita através da orientação à objetos, ou seja, classes, métodos, propriedades e relações entre esses objetos/recursos gerenciáveis. Isso tudo representa a interface pela qual a aplicação interage com os recursos gerenciáveis por meio da linguagem *VBScript*, como visto na seção anterior.

## **4. O Protótipo**

O protótipo de gerência WMI (ProGerWMI) é constituído de vários arquivos HTA's, que são arquivos que renderizam HTML em um ambiente diferente do navegador. Entre estes arquivos, há o principal contendo um menu que permite a execução de todos os outros responsáveis pelas operações. Entretanto, cada arquivo pode ser executado de maneira independente e na quantidade de instâncias desejada. Isso é útil para se fazer comparações entre os dados coletados em mais de uma máquina e/ou entre dados filtrados de formas diferentes entre as instâncias. No menu inicial há uma opção para se alterar o usuário que irá executar a operação, ou seja, conectar-se ao serviço WMI da

máquina “alvo”. Assim, é possível gerenciar máquinas em qualquer domínio da rede, bastando, para isso, efetuar o *logon* com o usuário apropriado. É importante ressaltar que, o usuário deve possuir permissão de administrador na máquina que se deseja gerenciar (máquina “alvo”). As credenciais de usuário só podem ser usadas se as operações forem iniciadas a partir do menu principal, de onde se efetuou o *logon*. A Figura F2 demonstra a tela principal da referida aplicação.



**Figura F2: Tela Principal do ProGerWMI**

Em função dos dados de autenticação serem adquiridos e armazenados na tela principal é necessário transferi-los para cada tela que é aberta a partir do menu inicial. Para isso, a função da Listagem L4 é utilizada.

```
01 sub Executar(opr)
02   Set Show = createObject("WScript.Shell")
03   Show.Run opr & " *"&domainUser & " *"&passWord
04   Set Show = nothing
05 end sub
```

**Listagem L4: Função de transferência dos dados de autenticação**

A função da Listagem L4 utiliza o método Run do objeto Shell (linha 03) para abrir cada arquivo, que é passado como parâmetro através da variável *opr*. Na chamada desse método, além do nome do arquivo são passados também o domínio e o nome de usuário, para que em cada janela filha a função apresentada na Listagem L5 realize a conexão com as credenciais fornecidas pelo usuário de nível “Administrador”. Essa forma de transferência de dados não seria possível caso o ambiente utilizado fosse o próprio navegador, pois não há como se ter acesso à linha de comando de uma janela do *browser*.

```
01 function conecta()
02   collCmdLn = split(listcomp.commandLine, " *")
03   if UBound(collCmdLn) > 0 then
04     user = collCmdLn(UBound(collCmdLn)-1)
05     passWd = collCmdLn(UBound(collCmdLn))
06   end if
```



```

07  pc = document.F_ListComp.txtEstacao.value
08  Set objConex = createObject("WbemScripting.SWbemLocator")
09  Set conectador = objConex.ConnectServer(pc,"root/cimv2"_
10  ,user,password)
11  Set objConex = nothing
12  set conecta = conectador
13 end function

```

#### Listagem L5: Função de conexão à máquina “alvo”

Todos os arquivos que realizam as operações possuem a função representada na Listagem L5, a qual será invocada sempre que houver uma solicitação de execução de uma operação qualquer. Os dados de *logon* são extraídos da linha de comando (linhas 02 a 06) e passados para o método `ConnectServer` do objeto `SwbemLocator` (linha 09), que retorna um objeto `SWbemServices`. Esse método realiza a conexão utilizando os dados dos parâmetros e, caso os mesmos sejam vazios, os dados da máquina e usuário corrente são utilizados.

As funcionalidades escolhidas e desenvolvidas no protótipo tiveram como base as necessidades e dificuldades encontradas para a gerência dos laboratórios de informática do CEULP/ULBRA. Assim, o ProGerWMI dispõe das seguintes funcionalidades:

- **Exibir Memória:** mostra informações sobre a memória física, arquivo de paginação e memória virtual de uma máquina;
- **Listar Processos:** A listagem de processos exibe um relatório, bem mais completo que o do gerenciador de tarefas padrão do *Windows*, com todos os processos em execução em uma estação. Sendo que é possível, também, realizar a finalização de um processo pelo identificador ou nome do mesmo;
- **Criar/Finalizar Processo:** Essa funcionalidade permite que o usuário crie um processo ou inicie um programa em uma estação local ou remota. Ou também, assim como na funcionalidade anterior, finalize um processo;
- **Listar Compartilhamentos:** Essa funcionalidade é parecida com a listagem de processos. Exceto que os dados exibidos aqui são acerca dos compartilhamentos;
- **Criar/Finalizar Compartilhamento:** Nessa funcionalidade é possível se criar compartilhamentos locais ou remotos, bem como a excluir os mesmos;
- **Configuração de Usuário/Estação:** Possibilita que o administrador tenha acesso à diversas informações acerca da conta de um usuário (nome de usuário, data de expiração da senha, etc) ou acerca das configurações de rede de uma estação (endereço IP, *Gateway*, etc ).
- **Listar/Configurar Serviços:** Essa é uma das operações mais completas, pois dá ao administrador uma noção exata de como estão se comportando os serviços de uma estação, além de possibilitar que esses sejam alterados conforme a necessidade. A grande facilidade dessa interface é permitir inúmeras filtragens, através de combinações dos dois principais atributos dos serviços, tipo de inicialização e estado;

- **Listar Espaço em Disco:** Aqui, o administrador tem a possibilidade de obter um relatório de quais estações em um laboratório estão com espaço livre em disco abaixo de um dado limite. Essa funcionalidade possui uma diferença, em relação as anteriores, na forma de gerenciamento: até então somente uma estação era “alvo” das operações, entretanto, essa e a seguinte, permitem que todas as estações de um laboratório sejam “alvo” de gerenciamento simultaneamente;
- **Desligar:** Por fim, tem-se a operação que permite realizar o desligamento, *logoff* ou reinicialização de uma estação ou todo um laboratório.

## 5. Conclusão

Este trabalho teve como objetivo apresentar aos administradores de redes um protótipo de uma ferramenta de gerência, que, além de ter funções úteis na administração dos recursos de uma rede, mostra a possibilidade de se criar ferramentas de gerência na plataforma *Windows* de uma forma bastante simples e com custo baixo, isto é, utilizando-se somente recursos já disponíveis no sistema operacional. Apesar de alguns dos recursos disponíveis não terem sido trabalhados no protótipo, como por exemplo, a utilização de eventos para a monitoração e disparo de ações, o seu desenvolvimento pode ser encarado como um ponto de partida para que os administradores criem suas próprias ferramentas de gerência. A vantagem de tal abordagem está na possibilidade de se controlar o escopo das operações que a ferramenta executará. Isso fará com que apenas operações pertinentes às necessidades de gerência estejam presentes na ferramenta.

## 6. Referências Bibliográficas

- DMTF. (2005) “DMTF - Web-Based Enterprise Management (WBEM)”, Disponível em: <http://www.dmtf.org/standards/wbem/>. Acesso em: 20 set. 2005.
- MSDN. (1999) “Managing Windows with WMI”, Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwm/html/mngwmi.asp?frame=true>. Acesso em: 20 set. 2005.
- MSDN. (2005a) “Introduction to HTML Applications (HTAs)”. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/workshop/author/hta/overview/htaoverview.asp?frame=true>. Acesso em: 22 set. 2005.
- MSDN. (2005b) “What Is VBScript?”. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vbswhat.asp?frame=true>. Acesso em: 22 set. 2005.
- MSDN. (2005c) “Windows Management Instrumentation (WMI)”. Disponível em: [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/anch\\_wmi.asp?frame=true](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/anch_wmi.asp?frame=true). Acesso em: 23 set. 2005.

- MSDN. (2005e) “Windows Management Instrumentation”. Disponível em:  
[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/about\\_wmi.asp?frame=true](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/about_wmi.asp?frame=true). Acesso em: 23 set. 2005.
- Specialski, E. S. (2002) “Gerência de Redes de Computadores e de Telecomunicações”, UFSC, Brasil.
- W3C. (2005) “W3C HTML Home Page” Disponível em: <http://www.w3.org/MarkUp/>. Acesso em 20 set. 2005.