Servlets e COM para a Visualização de Dados Geográficos na Web

ARLINDO CARDARETT VIANNA[1,2]
ANSELMO CARDOSO DE PAIVA[2,3]
MARCELO GATTASS[2]

[1] Universidade Estácio de Sá - Campus Barra
Av. Prefeito Dulcídio Cardoso, 2900,
22631-021 Rio de Janeiro, RJ, Brasil
acv@estacio.br

[2] PUC-Rio – Pontifícia Universidade Católica do Rio de Janeiro
TeCGraf – Grupo de Tecnologia em Computação Gráfica
Rua Marquês de São Vicente, 225
22453-900 – Rio de Janeiro, RJ
{acv,paiva, gattass}@tecgraf.puc-rio.br

[3] UFMA – Universidade Federal do Maranhão
Centro-Tecnológico - Departamento de Informática
Campus do Bacanga, S/N, 65000-000 – São Luís-MA
paiva@ufma.br

Abstract. This paper presents an ongoing study on the viability of using Java servlets and COM components for developing dynamic web sites containing geographic data. The maps are presented using TWF (TeCGraf Web Format). The technologies employed are discussed and the context in which they are useful is described.

Resumo. Este artigo apresenta um estudo em andamento sobre a viabilidade do uso de Java *servlets* e objetos COM para a criação de páginas dinâmicas que envolvam dados geográficos. Os mapas apresentados utilizam o formato TWF (TeCGraf Web Format). Faz-se uma abordagem das tecnologias utilizadas e apresenta-se uma descrição do contexto no qual elas são úteis.

1 Introdução

A popularização da Internet, o advento da *Web* e o crescimento da importância dos Sistemas de Geoprocessamento em uma série de atividades apresentam novas perspectivas para as pessoas que necessitam utilizar dados geográficos. A união desses três fatores torna possível a interação visual com os dados geográficos e uma maior acessibilidade a eles.

O usuário pode solicitar a visualização de mapas e gerar gráficos a partir dos dados geográficos armazenados, os quais podem ser atualizados em tempo real junto com a base de dados. Em razão da expansão da Internet é possível trabalhar com os dados de praticamente qualquer lugar.

As diferentes maneiras utilizadas para publicar dados geográficos na Internet, em geral, recaem no modelo clássico da Internet (cliente/servidor). Essa arquitetura é composta de um cliente com capacidade de exibição de dados geográficos e um programa de servidor de geoprocessamento sendo executado juntamente com o servidor *Web* para responder às solicitações sobre os dados geográficos. A Figura 1 representa essa arquitetura. Tipicamente o cliente é um navegador (como Netscape ou IExplorer) que realiza solicitações a um servidor *Web* utilizando o protocolo HTTP (*Hypertext Transfer Protocol*). Quando o cliente realiza a solicitação de um mapa ou outra operação sobre os dados geográficos, o servidor *Web* a repassa para o servidor de geoprocessamento, que atende à solicitação e envia a resposta para o cliente através do servidor *Web*. A comunicação entre os dois servidores é feita, em geral, através de interfaces como CGI (*Common Gateway Interface*), ISAPI (*Internet Server Application Program Interface* – Microsoft) e NSAPI (API do servidor Netscape).

Conforme descrito em Plewe[1], existem algumas maneiras básicas para publicar o dado geográfico na *Web*:

- copiando a base de dados geográficos completa ou parcialmente no cliente;

- exibindo mapas estáticos em páginas HTML (*Hypertext Markup Language*);
- possibilitando ao cliente navegar na base de metadados;
- permitindo a navegação em mapas dinâmicos;
- permitindo ao cliente a realização de consultas e análises sobre a base de dados no servidor, assim como a navegação em mapas dinâmicos.

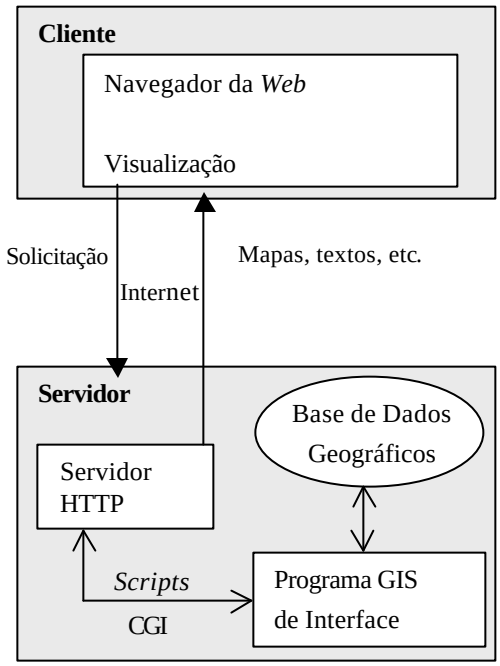

Figura 1 Arquitetura Típica para a Publicação de Dados Geográficos na *Web*.

Entre as opções mais utilizadas para a publicação de dados geográficos na *Web* podemos citar o SpringWeb [2] MapGuide (AutoDesk) [3], o GeoMedia WebMap (Intergraph) [4] e o MapObjects Internet Map (ESRI) [5].

O SpringWeb é um representante da primeira maneira para a publicação de dados geográficos na *Web*. Trata-se de um *applet* Java que roda no cliente e precisa da realização de uma cópia local da base de dados, apresentando então um nível razoável de interação com esses dados. Entre as funções disponibilizadas estão operações de visualização como *zoom*, *pan* e navegação sobre os mapas, além da possibilidade de consulta aos atributos de um objeto geográfico específico e a uma tabela com o conjunto de dados de uma determinada categoria de objetos.

O MapObjects Internet Map apresenta como desvantagem principal o fato de publicar os mapas como imagens, em geral nos formatos JPEG ou GIF, e suportar somente as bases de dados nos formatos proprietários da ESRI. Embora o usuário possa realizar consultas sobre os dados, em razão dos mapas estarem em formato *raster* não é possível navegar através de entidades gráficas.

O MapGuide utiliza um formato proprietário para os mapas a serem visualizados chamado SDF. No entanto, existe uma ferramenta para importar dados de uma série de outras bases, como ArcView e MapInfo, que possibilita a navegação sobre os mapas, mas sem permitir gerá-los dinamicamente.

O Geomedia WebMap apresenta como principal vantagem a capacidade de publicar dados na *Web* de diferentes formatos, como MGE e ArcView Shapefiles, entre outros. Neste programa os mapas são exibidos no formato ActiveCGM para clientes MSWindows ou como JPEG para clientes em outros sistemas operacionais. Uma de suas principais desvantagens advém da considerável quantidade de programação em *scripts* CGI necessária para construir uma aplicação completa. Esses *scripts* acessam objetos COM desenvolvidos como uma parte da versão completa do Geomedia, os quais realizam as operações sobre os dados geográficos e geram os arquivos para a exibição dos mapas. Esse sistema permite a geração dinâmica de mapas e a navegação através deles.

Todos esses programas têm em comum o fato de trabalharem utilizando CGI, ISAPI ou NSAPI como interface de comunicação entre o servidor de dados geográficos e o servidor de HTTP. Essas interfaces apresentam uma série de desvantagens, que serão discutidas em seguida.

Neste trabalho será apresentado um estudo sobre a viabilidade da utilização de Java *servlets* e componentes COM para o desenvolvimento de *sites* exibindo dados geográficos. Nesse estudo pretende-se utilizar o formato TWF (*Tecgraf Web Format*), desenvolvido por Gattass *et all*.[6], para a exibição dos mapas. A implementação da arquitetura aqui proposta se encontra em desenvolvimento no TeCGraf/PUC-Rio. Inicialmente serão apresentadas as tecnologias utilizadas, destacando suas principais vantagens e a razão de sua adequação para o tratamento do problema. Em seguida é fornecido o contexto de utilização dessas tecnologias para a publicação de dados geográficos na *Web*.

2 Java Servlets

Um *servlet* é um módulo (programa escrito em Java) que é carregado dinamicamente para atender às solicitações de um servidor *Web*, ou seja, é uma extensão acrescentada ao servidor que aumenta a sua funcionalidade. Os servidores *Web* respondem às solicitações dos usuários, geralmente, usando o protocolo HTTP através do envio de documentos escritos em HTML.

Em linhas gerais, os *servlets* estão para os servidores *Web* assim como os *applets* (componente de um navegador gráfico e não uma aplicação) estão para os navegadores, com a diferença que os *servlets* não possuem interfaces gráficas.

Os *servlets* podem ser carregados em diversos servidores, pois a API utilizada para escrevê-los usa apenas o ambiente da Máquina Virtual do servidor. O protocolo HTTP é o mais utilizado para a comunicação com o servidor. Os *servlets* podem ser usados para o processamento de formulários (HTML), interagir com bancos de dados, ou de uma maneira geral atuar como uma camada intermediária em uma arquitetura de três camadas, como apresentado esquematicamente na Figura 2.

As principais alternativas para a montagem de páginas dinâmicas são as seguintes: CGI, APIs proprietárias, ASP e *servlets*.

CGI é um padrão de interface utilizado entre servidores HTTP e programas para se comunicarem. O servidor executa um programa (chamado *script*) cujo resultado é então transmitido ao cliente. Normalmente, um *script* CGI gera como resultado uma página HTML para ser exibida no navegador. Os programas CGI são executados na máquina onde está localizado o servidor *Web*. Eles recebem os dados através de variáveis de ambiente e "entrada padrão" e os transmitem de volta ao servidor através da "saída padrão". Podem ser escritos em qualquer linguagem (C/C++, PERL, TCL, CGILua, entre outros). A principal desvantagem de se utilizar *scripts* CGI é que para atender a cada requisição do usuário o servidor *Web* precisa criar um novo processo (cada processo produz uma nova conexão com o banco de dados utilizado e o servidor *Web* tem que esperar até que os resultados lhe sejam enviados), o que resulta numa baixa eficiência. Outro ponto negativo é com relação à segurança, uma vez que os arquivos (*scripts*) não ficam inteiramente protegidos, pois devem ser armazenados nos sub-diretórios CGI-bin do servidor. Mais uma grande desvantagem está relacionada à baixa taxa de reutilização do código, em função das linguagens usadas para a programação. Para criar um novo módulo, a única possibilidade de reutilização é o "copiar-colar".

Muitos servidores *Web* incluem APIs proprietárias que expandem a sua funcionalidade. Os exemplos mais difundidos são a NSAPI (API do Servidor da Netscape), na qual os programadores podem criar módulos binários que irão acrescentar e/ou substituir elementos para autenticação, autorização ou geração dinâmica de conteúdos, e ISAPI (*Internet Server Application Program Interface* – Microsoft), que tem como princípio básico criar uma DLL que é carregada no servidor quando o HTTP é inicializado e permanece instalada enquanto for necessária. Esta DLL irá gerenciar a conexão com a base de dados sem a necessidade de se criarem novas conexões, como ocorre com o CGI. O grande problema em adotar tais soluções é que elas são proprietárias, perdendo-se a portabilidade. Outro problema é que geralmente estas aplicações são desenvolvidas utilizando-se linguagens de programação como C ou C++, que podem interferir no bom funcionamento do servidor *Web*.

Figura 2 Arquitetura de execução de um *servlet*.

A solução da Microsoft para a criação de páginas dinâmicas é o ASP (Active Server Pages). Ele fica embutido nas páginas HTML, não sendo pré-compilado. O servidor *Web* processa um arquivo utilizando uma DLL (ASP.DLL) que irá interpretar os comandos ASP embutidos na página HTML. O grande problema desta solução é que ela está ligada a um tipo particular de servidor – o Internet Information Server (IIS) – ou a alguma ferramenta que consiga fazer a tradução do ASP antes de enviar a resposta para o cliente.

Segundo Goodwill[7], as principais razões que indicam o uso de Java *servlets* são as seguintes:

- Eficiência: a inicialização de um *servlet* é feita apenas uma vez, ou seja, após ele ser carregado pelo servidor *Web*, as novas solicitações são apenas chamadas do método de serviço.
- Persistência: os *servlets* podem manter o estado entre solicitações.
- Portabilidade: eles são escritos usando Java e, por este motivo, são portáteis, bastando utilizar o JVM (Java Virtual Machine) do servidor, sem fazer uma alteração sequer no código-fonte.
- Robustez: Java possui um método bem definido para o tratamento de erros que minimiza a perda de memória por alocações indevidas (*garbage collector*).
- Segurança: os *servlets* por si só herdam toda a segurança que é peculiar a um servidor *Web*, porém os Java *Servlets* também podem contar com o Java Security Manager.
- Reutilização: como os *servlets* são escritos em Java, eles possuem as vantagens da utilização de uma

linguagem orientada a objetos, como a possibilidade de reutilização.

3 TWF

TWF (*Tecgraf Web Format*) é um formato de armazenamento de dados criado para representar um desenho que contenha linhas, regiões, textos e imagens que deverá ser transmitido via *Web*.

O formato de arquivo TWF foi proposto por Ferreira *et all*.[8] com o objetivo de obter um arquivo vetorial que atendesse aos requisitos propostos pelo W3C (World Wide Web Consortium)[9]. Esses requisitos definem um arquivo vetorial com capacidade de organização dos elementos em camadas, exibição de dados *raster*, tratamento de sistemas de coordenadas e possibilidade de interação com o usuário para operações de *zoom*, *pan*, seleção de objetos, tratamento de eventos, entre outras. Essa proposta foi estendida por Gattass *et all*.[6] para suportar a exibição de mapas na *Web*.

Um arquivo TWF é composto de um cabeçalho e uma série de camadas. Cada camada é composta por funções e cada função possui um único identificador e um número variável de argumentos.

A representação de mapas segundo esse formato baseia-se em dois processos: codificação dos elementos geométricos do mapa em um arquivo e interpretação do arquivo.

O processo de codificação realiza as seguintes operações: quantização das coordenadas, simplificação das polilinhas, codificação das coordenadas e compressão. Por outro lado, o processo de interpretação do arquivo consiste de somente dois passos: descompressão e interpretação das primitivas.

Existem duas ferramentas para a geração de arquivos TWF: uma para programadores e outra para usuários finais. A ferramenta para programadores é uma biblioteca de funções escrita em ANSI-C. Para usuários finais foi desenvolvida uma extensão do Geomedia, baseada na tecnologia COM, que exporta dados no formato TWF, mapeando cada *feature* em uma camada do arquivo.

Estão disponíveis dois visualizadores para esse formato de arquivo: um *applet* baseado na plataforma Java 1.2[10] e um *plug-in* desenvolvido em C com OpenGL[11].

Entre as principais vantagens desse formato de arquivo podemos enumerar:

- formato vetorial de arquivo, permitindo operações de visualização no cliente sem acessos ao servidor;
- geração de arquivos menores do que a maioria dos outros formatos disponíveis, conforme demonstrado em Gattass *et all*.[6].

4 COM e DCOM

Os componentes de *software* são uma evolução natural no desenvolvimento de *software* orientado a objetos, possibilitando o isolamento de pedaços de uma aplicação em componentes separados, os quais podem ser reutilizados por outras aplicações.

O *Component Object Model* (COM), da Microsoft, (Rogerson[12]) define um padrão binário para a integração de componentes, permitindo que objetos escritos em uma linguagem possam ser acessados a partir de outra.

A motivação da Microsoft para criar o COM partiu da necessidade de se estabelecer um padrão que permitisse que objetos criados por diferentes desenvolvedores pudessem interagir uns com os outros, de forma efetiva.

Esse modelo utiliza todos os conceitos básicos de orientação a objetos: encapsulamento, herança e reutilização. De uma forma geral, um objeto COM responde a uma série de métodos que serão utilizados pelos clientes.

As aplicações interagem umas com as outras ou com o sistema através de um conjunto de chamadas de funções ou métodos, que são conhecidas como interface. Esta deve ser desenvolvida de forma que possa ser reutilizada em vários contextos.

O modelo DCOM (*Distributed Component Object Model*) é uma extensão do modelo COM para suportar a comunicação entre objetos em computadores distintos, seja através de uma rede ou mesmo da Internet. Com o DCOM, a aplicação pode ser distribuída em vários locais de acordo com a conveniência do desenvolvedor e/ou do usuário. É importante ressaltar que o DCOM cuida dos detalhes de protocolos de rede de mais baixo nível, liberando o desenvolvedor para que este centralize seus esforços na busca por soluções mais rápidas e eficientes.

5 Arquitetura Proposta

Na arquitetura proposta, esquematizada na Figura 3, temos um cliente (navegador) se comunicando com um servidor *Web* através de requisições no protocolo HTTP, as quais são respondidas pelo servidor com o envio para o cliente de um arquivo HTML com os dados solicitados.

Em um ambiente *Web* para geoprocessamento é desejável que as páginas HTML sejam geradas dinamicamente em uma aplicação (Servidor de Dados Geográficos) integrada ao servidor *Web*. Existem vários mecanismos para realizar a integração entre essa aplicação e o servidor, como já foi visto. Na arquitetura proposta está sendo utilizado o mecanismo de Java *Servlets*. Com ele escrevemos um programa Java que é chamado pelo servidor *Web* para tratar as solicitações enviadas pelo cliente. Entre outras vantagens, os *servlets* possibilitam o acesso às facilidades de interconexão oferecidas por Java e a bases de

dados relacionais através do JDBC, facilidades para reutilização e modularidade e a perspectiva de ganhos de desempenho em razão de não precisar criar um novo processo para atender cada uma das solicitações do cliente. Assim, o *servlet* funcionará como a aplicação que atenderá às solicitações dos clientes, através do servidor *Web*, e gerará dinamicamente as páginas HTML para a publicação dos dados geográficos, exercendo um papel de mediador entre o cliente e os dados geográficos.

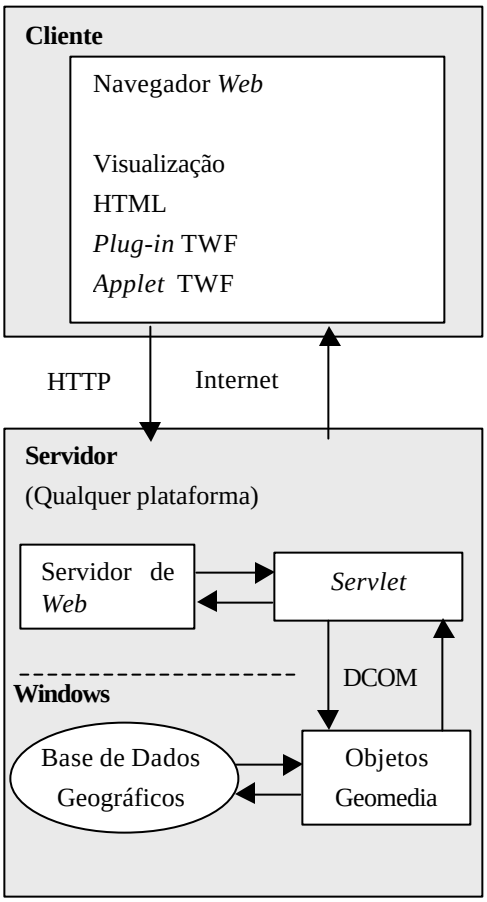

Figura 3 Arquitetura Proposta.

Para acessar os dados geográficos e responder a solicitações como a geração de um mapa ou a realização e exibição de uma consulta espacial ou de uma operação de interseção entre objetos geográficos, o *servlet* precisa de uma biblioteca de objetos que lhe permita realizar essas operações. Essa biblioteca é disponibilizada pelo Geomedia como um conjunto de objetos COM, os quais expõem uma parte de sua funcionalidade. Como principal vantagem da utilização desses objetos podemos citar o acesso a diferentes bases de dados, como MGE e ArcView.

Para a transmissão e exibição dos mapas no cliente foi escolhido o arquivo TWF, por se tratar de um formato vetorial, compactado e que disponibiliza operações de visualização (*pan*, *zoom*, etc.) no cliente. Além disso, esse formato vetorial é aberto, estando disponível na *Web* tanto a proposta do formato como *plug-ins* e *applets* para visualizá-lo, o que permite ter clientes em qualquer sistema operacional. A biblioteca para a geração desses arquivos, a partir de bases acessáveis pelos objetos Geomedia, está disponível como um objeto COM.

Um importante aspecto nesta arquitetura que está sendo proposta é a comunicação entre o *servlet* escrito em Java e os objetos COM, possivelmente rodando em máquinas distintas. A Microsoft estendeu sua implementação da Máquina Virtual Java para permitir uma integração automática entre Java e COM. Assim, um objeto COM pode ser chamado diretamente de Java e os objetos implementados em Java são disponibilizados como objetos COM. Essa solução possui a desvantagem de exigir que o *servlet* esteja rodando uma Máquina Virtual Java, que não é o padrão.

Na nossa proposta optamos por realizar essa comunicação através do J-Integra[13], uma ferramenta de integração desenvolvida pela Linar Ltda. Ela se comunica com os objetos COM utilizando o DCOM (*Distributed COM*), que se baseia em uma camada de RPC, a qual realiza chamadas TCP/IP para efetivamente completar a comunicação. Em resumo, no nível mais baixo o J-Integra usa as classes padrões de acesso à comunicação em rede oferecidas pela linguagem Java, o que garante a portabilidade da solução.

Para o programador do *servlet*, a solução com o J-Integra possibilita que os componentes COM sejam tratados como objetos Java puros, apresentando as propriedades, os métodos e os eventos do objeto COM como propriedades, métodos e eventos Java.

A partir da leitura de bibliotecas que contenham informações das classes e interfaces dos objetos COM, uma ferramenta distribuída com o J-Integra – com2java – gera os *proxies* necessários para a ligação entre COM e Java, ou seja, arquivos que contenham as classes em Java equivalentes aos componentes COM. Podemos entender estas classes geradas como um *wrapper* para os objetos COM. Essas bibliotecas podem estar explicitamente definidas em arquivos com extensão *olb* ou *tlb*, e algumas vezes as informações estão escondidas em arquivos executáveis ou em DLLs.

No desenvolvimento deste sistema já foi implementado um *servlet* que, sem a utilização de objetos COM, gera páginas HTML com mapas previamente gerados no formato TWF. A Figura 4 mostra uma página produzida com esse *servlet*, que possui capacidade de conexão direta com o

banco de dados associado ao mapa para a realização de consultas convencionais a bancos de dados. Atualmente está sendo desenvolvido o empacotamento dos objetos COM em classes Java, utilizando a ferramenta com2java do J-Integra. Isso é necessário para a geração dinâmica dos mapas a partir da base de dados acessável pelo Geomedia.

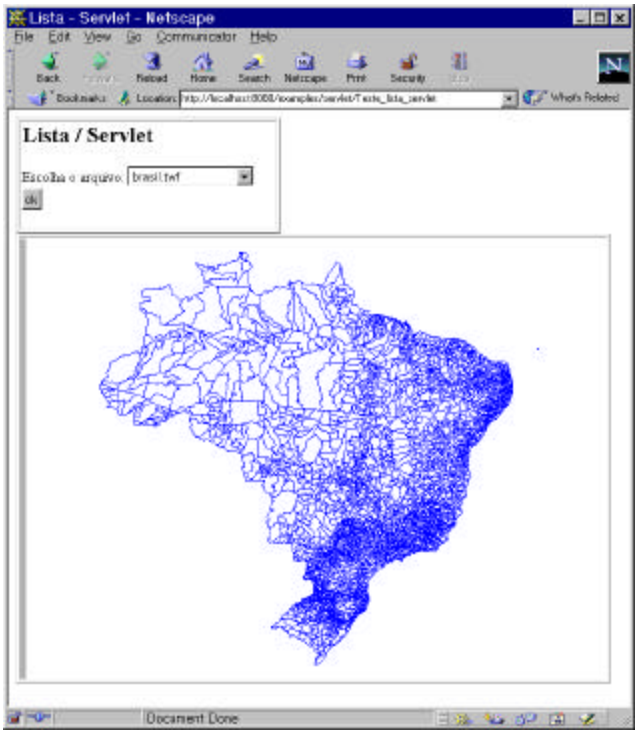

Figura 4 Exibição de Mapa em TWF com *servlet*.

Agradecimentos

Tecgraf é um laboratório financiado principalmente pela PETROBRÁS.

Referências

[1] Plewe, B. *GIS ONLINE: Information Retrieval, Mapping, and the Internet*. ONWORD Press, 1997.

[2] SpringWeb.
www.dpi.inpe.br/springweb/springweb.html

[3] MapGuide. www.mapguide.com

[4] GeoMedia Web Map.
www.intergraph.com/software/geo_map/

[5] MapObjects Internet Map Server.
www.esri.com/base/products/internetmaps/

[6] Gattass, M.; Ferreira, C. C. F.; Vilar, A. S.; Glasberg, M. "Efficient Map Visualization on the Web" . Geoinfo'99, Campinas, São Paulo, 1999.

[7] Goodwill, J. *Developing Java Servlets – The Authoritative Solution*. Sams Publishing, 1999.

[8] Ferreira, C. C. F.; Gattass, M.; Figueiredo, L. H. "Um Estudo Sobre Arquivos Vetoriais para a Visualização de Mapas na Web" . GIS Brasil 99 – V Congresso e Feira Para usuários de Geoprocessamento da América Latina, Salvador, Bahia, 1999.

[9] World Wide WebConsortium Requeriments.
www.w3.org/Graphics/ScalableReq

[10] TWF Applet. www.tecgraf.puc-rio.br/twf/applet/

[11] TWF Netscape Plug-In.
www.tecgraf.puc-rio.br/twf/plugin/

[12] Rogerson, D. *Inside COM*. Microsoft Press, 1997

[13] J-Integra™. www.linar.com