Utilização de SVG na Visualização de Sinópticos

Filipe Marinho[1], Paulo Viegas[1], João Correia Lopes[23]

[1] EFACEC Sistemas de Electrónica, Rua Eng. Frederico Ulrich, Apartado 3078, 4471-907 Moreira Maia.
http://www.efacec.pt/
{filipe.marinho,pviegas}@se.efacec.pt

[2] Faculdade de Engenharia da Universidade do Porto, R. Dr. Roberto Frias, 4200-465 Porto.
http://www.fe.up.pt/∼jlopes/
jlopes@fe.up.pt

[3] INESC Porto, R. Dr. Roberto Frias, 4200-465 Porto.
http://www.inescporto.pt/

Resumo O GENESys é um projecto desenvolvido pelo Departamento de Gestão de Redes, Unidade de Automação de Sistemas de Energia da *EFACEC*, em parceria com a EDP (*Energias de Portugal*), que visa a implementação de um sistema *SCADA/DMS* (*Supervisory Control and Data Acquisition/Distribution Management System*) que controle a rede eléctrica nacional.
A disponibilização do GENESys na *Web* traz grandes vantagens para a manutenção do sistema, mas para isso é necessário encontrar uma tecnologia que garanta uma interacção, usabilidade e desempenho semelhante à actual, que assenta em aplicações Java.
Neste artigo descreve-se a utilização da tecnologia *SVG* (linguagem em formato XML que descreve gráficos de duas dimensões) para representar sinópticos e permitir a interacção do utilizador do sistema GENESys. Com base num protótipo desenvolvido, foi feita uma avaliação desta tecnologia no sentido de verificar se cobria todos os requisitos para a visualização de sinópticos na Web.
Este estudo mostrou a viabilidade da utilização da tecnologia *SVG* para visualizar sinópticos e actualizá-los dinamicamente com a recepção de informação dos sistemas *SCADA/DMS*. Para além disso, verificou-se que as capacidades interactivas dos *SVG*s permitem enviar comandos e consequentemente actuar sobre o sistema.

1 Introdução

Os sistemas *SCADA* (*Supervisory Control And Data Acquisition*) são responsáveis pela recolha e análise de informação em tempo real. Actualmente estes sistemas recorrem às mais avançadas tecnologias de computação e comunicação para monitorizar e controlar estruturas ou equipamentos industriais dispersos geograficamente e recorrem a interfaces gráficas sofisticadas para tornar a interacção com o utilizador mais amigável.

Este tipo de sistemas é usado em várias indústrias: telecomunicações, águas e saneamento, energia, gás, combustíveis e transportes. Hoje em dia estes sistemas são distribuídos, com utilizadores e dispositivos dispersos, dispositivos estes que poderão falhar sem prejudicar o sistema, uma ou mais bases de dados e com suporte para um desenvolvimento contínuo em várias plataformas de computação.

Os sistemas DMS (*Distribution Management System*), surgiram como uma evolução dos sistemas *SCADA* usados na supervisão de redes eléctricas. Os sistemas *SCADA* monitorizam pontos (equipamentos) da rede que controlam, possibilitando também o seu controlo, mas não têm noção da interligação destes pontos. Por outro lado, os sistemas DMS dispõem de informação do modelo de conectividade da rede, que permite disponibilizar uma série de ferramentas que ajudam no controlo da rede eléctrica.

O GENESys é o resultado de um projecto conjunto entre a *EFACEC Sistemas de Electrónica S.A.* e a EDP (*Energias de Portugal*) para o desenvolvimento de um sistema *SCADA/DMS* de nova geração, no âmbito da execução de um projecto da EDP mais alargado, o projecto SIRED (*Sistemas Integrados para Redes Eléctricas de Distribuição*). O objectivo deste sistema é reduzir o número de centros de controlo da rede de distribuição nacional detidos pela EDP e aumentar a sua eficácia.

O GENESys monitoriza, em tempo real, todas as medidas relativas à rede eléctrica, tais como a corrente, a voltagem, a potência, etc, apresentando-as graficamente para que possam ser interpretadas pelo operador. Da mesma forma é possível, a qualquer momento, actuar sobre os equipamentos: mudar o estado do equipamento, gerir ordens de manobra, ordens de serviço, planeamento e estudo de procedimentos, alocação de recursos e de equipas de reparação.

Alguns sistemas *SCADA*, têm um elevado grau de complexidade, quer na sua instalação quer na sua manutenção, havendo necessidade de ter pessoal especializado nessas funções. Acompanhando a evolução e tendência das tecnologias de informação, pretende-se disponibilizar estes sistemas na *Web*, tendo como vantagens o acesso ilimitado através de um navegador *Web* e a limitação da necessidade de instalação e configuração só nos servidores do sistema.

Um dos módulos mais críticos na disponibilização do GENESys na *Web* é a visualização de sinópticos. Um Sinóptico é a representação gráfica do sistema, que no caso do GENESys é a rede eléctrica (ver figura 2). O transporte da visualização de sinópticos para a *Web* levanta algumas questões importantes. É preciso encontrar uma tecnologia que garanta uma interacção, usabilidade e desempenho satisfatórios em ambiente *Web*.

Neste trabalho é usada a tecnologia *SVG*, por forma a verificar se esta cobre todos os requisitos para a visualização de sinópticos na *Web*. Pretende-se que a representação do sinóptico em *SVG* permita interagir com o sistema GENESys, ou seja, permita alterar a representação dos símbolos consoante a informação recebida do sistema e controlar o estado dos símbolos através da interacção com o sinóptico.

Para além desta introdução, onde se caracterizou o problema abordado por este projecto, faz-se de seguida uma abordagem ao estado da arte e são des-

critos os trabalhos relacionados com a visualização de diagramas sinópticos de sistemas *SCADA* na *Web*. Na secção 2 é feita uma descrição da arquitectura física e tecnológica do protótipo desenvolvido. Na secção 3 faz-se a descrição da arquitectura que suporta o GENESys e das ferramentas utilizadas para o desenvolvimento do visualizador de sinópticos. Na secção 4 é detalhado o desenvolvimento do visualizador *SVG* e na secção 5 são descritos e analisados os testes de desempenho. Finalmente, na secção 6 são apresentadas as conclusões e os trabalhos futuros.

2 Estado da Arte e Trabalhos Relacionados

Nesta secção faz-se a introdução à realidade do sector da Automação de Sistemas, no âmbito deste artigo. Será feito um resumo das tecnologias que podem ser utilizadas na disponibilização de diagramas sinópticos na *Web* e das soluções existentes no sector nesse contexto.

2.1 Clientes Simples e Clientes Complexos

Numa abordagem baseada em clientes simples (*thin client*), a preocupação reside na utilização de tecnologias naturalmente suportadas pelos navegadores e que, por isso, não incorrem na necessidade de descarregamento de blocos de código pesados, tanto pelo seu tamanho, como pela capacidade computacional que necessitam [LFCJ02] [LSFH00] [SLN99] [ZPMD97].

Já numa abordagem baseada em clientes complexos (*rich client*) existe uma clara utilização da capacidade computacional do cliente, em oposição ao que acontece com as abordagens para clientes simples. Esta característica permite a implementação de blocos substanciais de código para a sua execução, podendo em alguns casos, limitar a utilização deste tipo de soluções [Fer03].

2.2 Solução Baseada numa Ligação VPN

Uma das soluções apresentadas por empresas do sector Automação de Sistemas para a disponibilização de sistemas *SCADA/DMS* na *Web*, é uma aplicação cliente baseada em *Java* (*rich client*) que permite o acesso a todas as funcionalidades do sistema *SCADA/DMS*. A aplicação cliente comunica com o sistema *SCADA/DMS* através de uma ligação VPN ou SSL à rede onde os servidores estão alocados. Esta aplicação não é desenvolvida para a *Web*, não havendo preocupações quanto ao tamanho desta pois é instalada *offline*. Nesta solução, a representação de sinópticos na *Web* é feita através de componentes *Java* incorporados na aplicação cliente.

2.3 Páginas HTML

Nesta solução utiliza-se HTML para a definição da interface, através da inclusão de imagens de diagramas geradas pelo servidor, que vão sendo actualizadas ao longo do tempo, através do refrescamento periódico da página.

A execução de controlos é suportada pela inclusão de formulários (ou diálogos gerados por *Javascript*) que resultam na execução de um CGI ou de uma *servlet* no servidor para actuação no processo. Um mecanismo semelhante poderia ser utilizado com vista à inclusão de facilidades de navegação: pedidos de ampliação e arrastamentos que resultavam na geração de imagens ampliadas ou deslocadas, conforme requerido. As animações poderiam ser conseguidas pela utilização de GIFs ou PNGs animados. No servidor haveria a necessidade de implementação de um sistema de geração de imagens, que ia servindo os pedidos recebidos com base na informação recolhida do sistema *SCADA*.

Os principais problemas desta solução são:

- A geração dinâmica de imagens "rasterizadas" (GIFs e PNGs) com animações é pesada e pouco eficiente;
- Há necessidade de carregamento total da página a cada pedido de refrescamento. Uma vez que o conteúdo de uma página poderia incluir uma ou mais imagens dinâmicas de grandes dimensões, a quantidade de informação trocada entre clientes e servidor poderia ser muito grande e, por isso, condicionada pela largura de banda do canal de transmissão;
- Há necessidade de uma taxa de refrescamento elevada. As características de tempo-real dos sistemas *SCADA* levam à necessidade de actualização frequente da informação visualizada. Este factor, em conjunto com a eventual existência de outros clientes que efectuam acessos simultâneos ao sistema, introduz uma enorme sobrecarga no motor de geração de páginas que, por serem complexas e de grandes dimensões, pode conduzir ao atraso na disponibilização da resposta.

2.4 Visualizador Remoto

Um visualizador remoto consiste numa aplicação que permite a visualização de interfaces gráficas que executam remotamente. Esta aplicação seria integrada no navegador (pela utilização de *Java applets* ou *ActiveX*) e comunicaria com um servidor no centro de comando através de um protocolo de transmissão de informação gráfica [LFCJ02] [LSFH00].

Esta é uma técnica conhecida como *Graphics Pipeline Interception* e o seu princípio de funcionamento consiste na existência de uma aplicação que intercepta os comandos gráficos ocorridos na máquina servidora, enviando a imagem resultante para visualização no cliente. Por seu turno, os eventos ocorridos no cliente são transmitidos ao servidor, que os executa localmente.

O problema associado aos sistemas de visualização remota é a largura de banda necessária para a transmissão de informação gráfica através da rede, mesmo que sejam utilizadas técnicas de compressão ou envio de comandos gráficos de alto nível, em vez do envio total da imagem a visualizar. Por outro lado, estes sistemas apresentam a importante vantagem de poderem ser integrados com sistemas existentes, através da intercepção das rotinas gráficas ao nível do sistema operativo.

Actualmente, empresas do ramo da Automação de Sistemas usam esta tecnologia para potenciar o acesso através da *Web* aos seus sistemas e consequentemente a visualização dos seus sinópticos. Como exemplo temos os *Terminal Services* disponibilizados pelos sistemas operativos *Windows Server 2000* e *Windows Server 2003* [Mic04] que permitem, através de um *thin client*, que qualquer máquina (PDA, PC, TabletPC, etc) com um qualquer sistema operativo (Windows, Unix, Mac OS, etc.) possa aceder através da *Web* qualquer aplicação baseada em Windows (neste caso o sistema *SCADA/DMS*) instalada nos servidores. A comunicação é baseada no RDP (*Remote Desktop Protocol*), o qual assegura a encriptação dos dados.

2.5 Componentes Especializados

Esta solução consiste na implementação de uma biblioteca de componentes gráficos *Java*, para a construção de clientes gráficos *SCADA*, que disponibilizam funcionalidades específicas deste tipo de aplicações. Estes componentes poderiam depois ser conjugados em *applets* que seriam descarregadas pelo navegador para execução local.

O desenvolvimento de componentes gráficos permite que grande parte das questões de interacção com o utilizador sejam resolvidas localmente sem necessidade de comunicação com as aplicações residentes no centro de comando, levando não só à diminuição do fluxo de informação entre clientes e servidores, mas também à possibilidade do aumento do grau de interactividade entre operadores e aplicação.

Esta solução permite que a informação transaccionada entre clientes e servidores seja informação de alto nível, que corresponda à própria semântica do problema, em vez de informação de baixo nível, como a presente nos sistemas de visualização remota, onde circulam imagens e directivas de desenho. Assim, os valores correspondentes a alterações de entidades poderão ser enviados directamente a clientes especializados, com conhecimento suficiente para proceder à sua adequada representação, seja por alteração da colocação, por animação de imagens ou simplesmente por apresentação textual do seu valor.

Em oposição a todas as vantagens que o desenvolvimento em *Java* fornece (multi-plataforma, utilização gratuita, segurança, Swing, etc) a utilização de componentes para a construção de interfaces *SCADA* na *Web* obriga à necessidade destes componentes serem relativamente pequenos, para que o processo de descarregamento das aplicações possa ocorrer num intervalo de tempo aceitável.

Outro problema desta abordagem reside na necessidade de implementação específica de componentes para os vários aspectos dos interfaces *SCADA*: sinópticos, listas, gráficos de tendências e alarmes. Mas, uma vez desenvolvidos, estes poderão ser utilizados para interligação com os sistemas existentes. Assim, a implementação de um sistema baseado em componentes gráficos implica a definição e desenvolvimento de uma arquitectura de software que permita estabelecer a comunicação entre os componentes e as aplicações pertencentes a um sistema *SCADA* [Fer03].

3 Utilização de *SVG* na Visualização de Sinópticos

Como vimos nas secções anteriores, pretendemos disponibilizar a visualização de sinópticos e suportar interactividade com o sistema através da *Web*. Das várias soluções tecnológicas ao nosso dispor, testámos a combinação das tecnologias *SVG* e *Java*, assentes na arquitectura distribuída BUS toolkit (ver secção 3.2). Este teste baseou-se no desenvolvimento de um protótipo que nos permitiu retirar as devidas conclusões.

Nesta secção faz-se a descrição detalhada da arquitectura que suporta o GENESys e das tecnologias e ferramentas utilizadas para o desenvolvimento do visualizador de sinópticos [Mar05a].

3.1 *SVG*

Scalable Vector Graphics é uma linguagem em formato XML que descreve gráficos de duas dimensões. Este formato normalizado pela W3C (*World Wide Web Consortium*) é livre de patentes ou direitos de autor e está *totalmente* documentado, à semelhança de outros W3C standards [Wor05b].

Sendo uma linguagem XML, o *SVG* herda uma série de vantagens: a possibilidade de transformar *SVG* usando técnicas como XSLT, de embeber *SVG* em qualquer documento XML usando *namespaces* ou até de estilizar *SVG* recorrendo a CSS (*Cascade Style Sheets*). De uma forma geral, pode dizer-se que *SVG*s interagem bem com as actuais tecnologias ligadas ao XML e à Web [IBM05].

Os *SVG* podem integrar objectos como formas vectoriais (rectângulos, círculos, "caminhos" constituídos por linhas rectas ou curvilíneas), transparências, filtros (efeitos de luz, sombras, etc), imagens e texto. Estes objectos também podem ser interactivos e dinâmicos, características essas que permitem a definição de animações que poderão ser despoletadas através de declarações embebidas nos *SVG* ou através de *scripting* [Wor05a] [Apa05b].

Esta tecnologia poderá ser utilizada na indústria, nomeadamente no suporte gráfico e interactivo dos sistemas *SCADA*, o que faz com que o operador possa monitorizar, controlar e operar sobre estes sistemas [Gar05].

3.2 *BUS* Toolkit

O *BUS* Toolkit é uma arquitectura publicador/subscritor desenvolvida para ser o "coração" do GENESys da *EFACEC* Sistemas Electrónicos, S.A..

Este tipo de sistemas apresenta alguns aspectos particulares que influenciam a filosofia da arquitectura, tais como: a sua complexidade, o facto de ser distribuída, de ter uma grande tolerância a falhas e uma enorme robustez e de ser configurável [MV01].

No ponto de vista do *BUS* um sistema é um conjunto de componentes (unidade básica de processamento). Estes componentes comunicam entre si através do Modelo *Push* ("impulso" ou "empurrão") e de acordo com o paradigma publicador/subscritor. Neste paradigma os componentes publicam sobre "assuntos" (tópicos) que interessam a outros componentes (e vice-versa). Toda a gestão de

quem publica e subscreve tópicos é feita por um componente especial, o Navigator. Este componente tem a função de informar quem publica e subscreve nos diferentes tópicos.

3.3 *BUS* — Canal Web

A possível manipulação de um sinóptico (ligado a um sistema *SCADA/DMS*) através de qualquer navegador *Web*, implica que a comunicação entre o protótipo e o *BUS* tenha de ser feita através da *Web*, usando um canal disponível na JFRK para o efeito [Gom03].

A finalidade do Canal Web é permitir que um qualquer componente que esteja a correr num navegador *Web* de uma máquina qualquer (numa *applet* ou então numa aplicação separada), comunique com o sistema central (onde estão localizadas as base de dados e principais componentes) através de um Canal *Web*, da mesma forma que comunicaria através de RMI ou IIOP (outros canais de comunicação entre componentes disponibilizados pela JFRK), caso estivesse a ser executado na rede local.

A principal razão da existência de um Canal *Web* é possibilitar a comunicação entre componentes em diferentes *Intranets*, o que obriga à passagem pelas *firewalls* que as protegem. Por norma a configuração das *firewalls* não permite a comunicação via RMI ou IIOP, sendo por isso necessário recorrer ao HTTP (*HyperText Transport Protocol*). A comunicação através do Canal *Web* baseia-se no acesso a um serviço *Web* que fará a ponte entre um componente a correr na *Web* e o GENESys (ver figura 1).

Foram elaborados alguns testes, para verificar a capacidade de comunicação através do canal *Web*. Verificou-se que este suporta o envio de 200 eventos por segundo, no teste elaborado dentro de uma *Intranet* com a velocidade de 100 KB/s. Já no teste efectuado com um acesso por VPN (*Virtual Private Network*) à *Intranet* através de uma ligação de 2 Mb/s, verificou-se um limite de 130 eventos por segundo, com a agravante de ter ligar um tempo excessivo no transporte dos eventos entre componentes do *BUS*, o que origina um "longo" período entre o momento em que os eventos são enviados de um componente e o momento em que são recebidos no outro [Mar05a] [Mar05b].

3.4 Batik *SVG* Toolkit

Batik é um conjunto de bibliotecas baseadas em *Java* que permitem o uso de imagens *SVG* (visualização, geração ou manipulação) em aplicações ou *applets*. O projecto Batik destina-se a fornecer ao programador alguns módulos que permitem desenvolver soluções especificas usando *SVG*. Entre esses módulos destacam-se [Apa05a]:

- *SVG* **Generator** — Modulo que permite a todas as aplicações *Java*, converter facilmente gráficos em *SVG* [Apa05c].
- *SVG* **DOM** — Este módulo permite criar uma representação para um qualquer *SVG* com o intuito de permitir a sua manipulação.

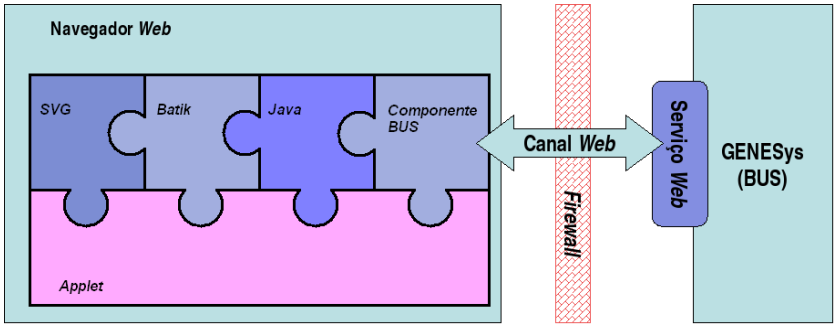

Figura 1. Arquitectura da Solução Proposta

– **J*SVG*Canvas** — Componente gráfico para visualização de *SVG*s e que permite a interacção com estes gráficos (aumentar ou diminuir, rodar, seleccionar texto, etc).

Temos então no Batik uma solução fácil para integrar o *Java* com *SVG*.

4 Visualizador de Sinópticos

Para avaliar as capacidades da tecnologia *SVG* na representação e visualização de diagramas sinópticos na *Web*, optou-se por desenvolver um protótipo que cumprisse os requisitos necessários à visualização de diagramas do GENESys. Este protótipo foi desenvolvido em *Java*, para que possa correr num navegador *Web* (através de uma *applet*) e utilizou-se o Batik *SVG* Toolkit para a visualização e manipulação dos gráficos *SVG*.

Numa fase inicial foram definidos alguns requisitos funcionais, que cobrem os aspectos necessários à visualização de sinópticos na *Web*:

– Criação de um protótipo que permita a interacção com símbolos no *SVG* e que despolete o envio de um evento para o sistema central (simulando assim a possibilidade de actuar sobre o sistema). Esta interacção é feita através de menus ligados ao clique do botão direito do rato.
– O protótipo também deverá receber eventos que mudem o estado dos símbolos e proceder em conformidade (mudar cor, mudar símbolo, etc). Este requisito permite simular a "animação" do diagrama de acordo com a realidade da respectiva rede eléctrica controlada pelo GENESys.
– Cada símbolo no sinóptico tem um menu correspondente. O conteúdo deste menu poderá ser configurado através de um ficheiro XML à parte.

Neste protótipo não foram implementados em *SVG* todos os componentes gráficos que constituem um sinóptico real (ver figura 2). A título exemplificativo foram definidos alguns símbolos para que se pudesse testar a tecnologia. Estes símbolos representam medidas, *switches* e disjuntores nos seus vários estados (ver figura 3).

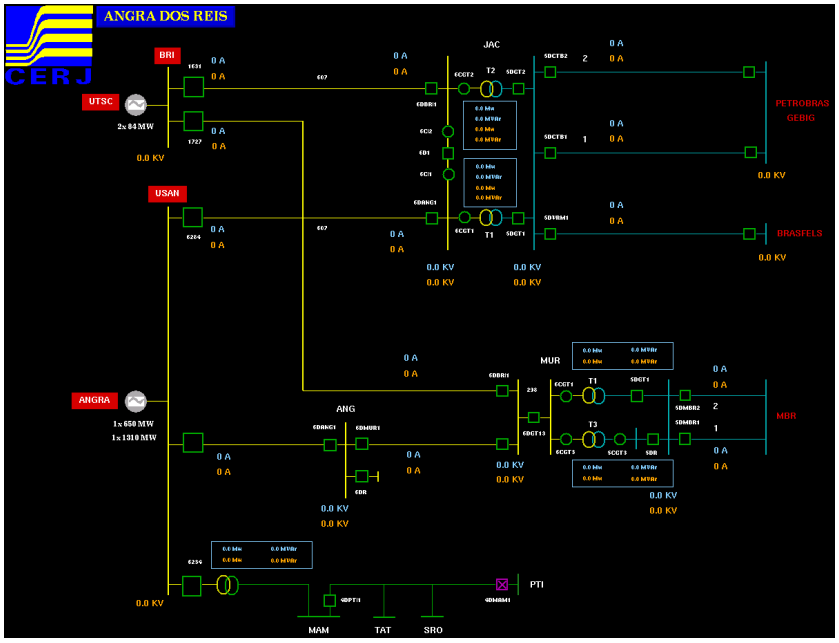

Figura 2. Sinóptico da Área de Angra dos Reis no Brasil

A principal funcionalidade deste protótipo é a de permitir a visualização de sinópticos a partir de qualquer localização, ou seja, através da *Web*, apoiando-se na arquitectura do Canal *Web* do *BUS*. Deste modo, o Visualizador *SVG* será uma *applet*, que correrá sobre um navegador *Web*.

Esta *applet* é um componente *BUS* (ver secção 3.2) que recebe e envia eventos através do Canal *Web*, o que lhe permite actualizar os sinópticos e actuar sobre o sistema (ver figura 4).

A ligação entre a aplicação em *Java* (*applet*) e os gráficos *SVG* é feita através da utilização da ferramenta Batik. Esta ferramenta disponibiliza um painel (*jSVGCanvas*), que permite visualizar e interagir com os gráficos *SVG* (fazer *zoom*, *scroll*, clicar sobre os objectos, etc).

A preocupação no desenvolvimento deste visualizador de sinópticos centrou-se nas funcionalidades, tendo assim uma interface gráfica simples, pensada para facilitar o estudo da tecnologia (ver figura 5)

5 Testes e Avaliação do Visualizador de Sinópticos

O protótipo desenvolvido demonstrou que a tecnologia *SVG* tem capacidades para construir e visualizar sinópticos dinâmicos, permitindo a interacção com o utilizador. Para fazer uma avaliação final sobre esta tecnologia, foi necessário efectuar alguns testes de desempenho. Para isso elaboraram-se alguns cenários que permitiram cobrir todas as áreas que se pretenderam testar: ocupação de

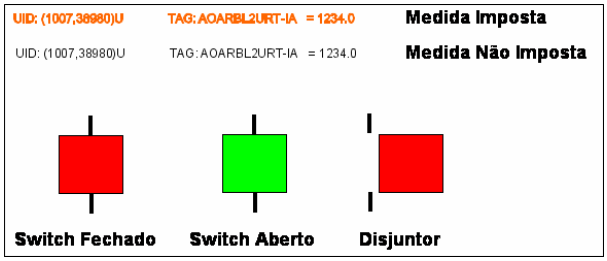

Figura 3. Simbologia Definida para o Protótipo

memória, utilização de *CPU*, desempenho no carregamento e na actualização dos *SVG*s, protótipo multi-plataforma, etc.

A elaboração destes cenários foi suportada pela utilização de dois computadores: um cliente e um servidor. O "servidor" representava o sistema central e permitia o descarregamento (para um navegador) da *applet* do visualizador de sinópticos. O "cliente" tinha a função de descarregar o visualizador e executar todos os testes necessários. O servidor tinha também como função a simulação de eventos que iriam alimentar o diagrama *SVG* no teste de actualização dos símbolos. Para o "cliente" foi usado um computador com um processador Pentium Centrino 1.5 Mhz com 512 MB RAM.

Alguns dos cenários foram efectuados duas vezes mas em modelos diferentes. Um dos modelos foi a elaboração dos testes com os dois computadores ligados à *Intranet* da *EFACEC Sistemas de Electrónica S.A.*, um segundo modelo foi ligar o "cliente" através da *Internet* à *Intranet* da *EFACEC Sistemas de Electrónica S.A.* e consequentemente ao "servidor".

No que diz respeito ao carregamento da *applet* nos vários navegadores testados (*Internet Explorer, Opera, Firefox*), os tempos foram semelhantes numa comparação entre navegadores e no geral aceitáveis, mesmo numa perspectiva de carregamento da *applet* através da *Internet*.

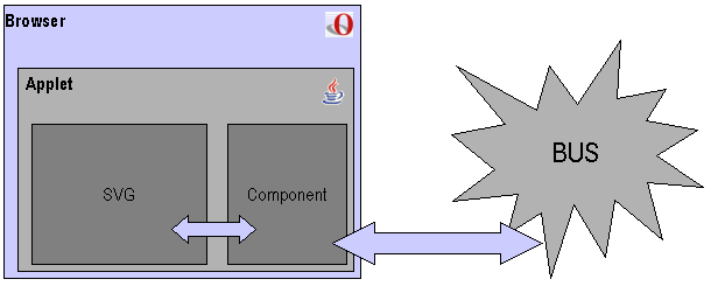

Figura 4. Arquitectura do Protótipo *SVG*

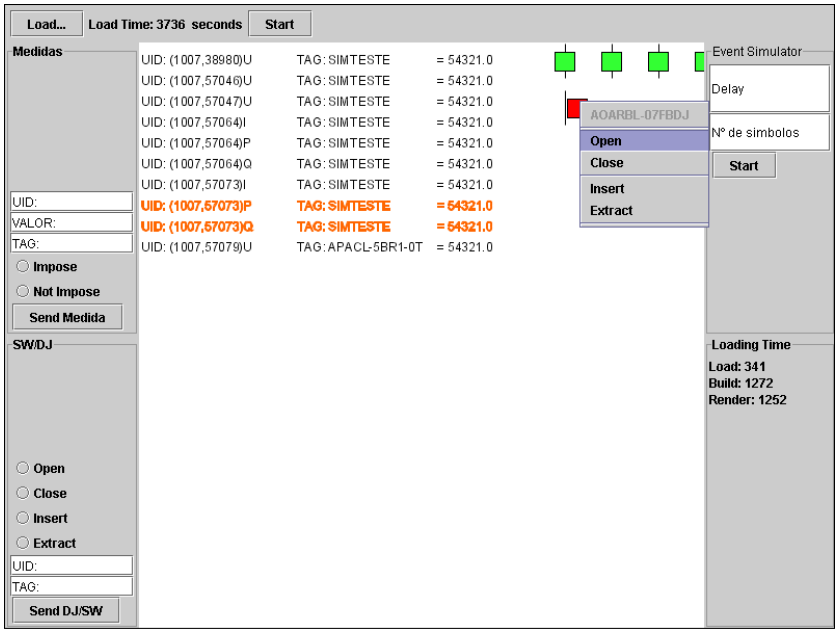

Figura 5. Interface do Visualizador de Sinópticos *SVG*

No carregamento dos diagramas *SVG* os tempos obtidos foram satisfatórios, verificando-se apenas que em diagramas de grande dimensão o carregamento era impossível devido à falta de memória virtual. Este problema foi resolvido com a passagem de parâmetros à VM (*Virtual Machine*) com o intuito de alocar mais memória virtual aos processos *Java*.

O cenário mais crítico foi o teste do desempenho do visualizador na actualização dos símbolos de acordo com o eventos que lhe iam chegando. Os resultados não foram satisfatórios, na medida em que o visualizador só conseguiu tratar eficazmente (tendo carregado um sinóptico de tamanho dentro dos limites impostos pela EDP (500 símbolos)) 8 eventos por segundo. Esta situação deve-se ao facto das operações de manipulação da árvore DOM, que representa o diagrama em memória, serem bastante pesadas (ver figura 6). Este limite coincide com uma utilização máxima do *CPU* do "cliente".

No mesmo cenário, mas correndo o visualizador fora da *Intranet* através de um computador ligado à Internet com uma ligação ADSL de 2Mbits por segundo, a eficácia no tratamento de eventos e consequente actualização dos símbolos manteve-se, tendo o mesmo diagrama carregado. O mesmo não aconteceu aquando a utilização de uma ligação GPRS de 56Kb por segundo, onde os resultados desceram para 4 eventos por segundo. Esta situação deve-se à reduzida largura de banda o que origina um sucessivo atraso no descarregamento dos eventos (agrupados em pacotes) e consequentemente um atraso no seu tratamento.

Figura 6. Resultado dos Testes de Desempenho na Actualização dos Símbolos

Verificados estes resultados, houve a necessidade de elaborar o mesmo teste, mas mudando a maneira de actualização dos símbolos. Em vez de substituir cada símbolos de acordo com os eventos recebidos, altera-se apenas uma propriedade do símbolo (mudando a cor, texto, etc). Esta versão do protótipo levou a resultados muitíssimo melhores, chegando a tratar 160 eventos por segundo, com o mesmo diagrama referido anteriormente. A conclusão retirada foi que a operação de substituir a representação de um símbolo do diagrama na árvore DOM é bem mais pesada que a operação de alterar apenas uma sua propriedade.

6 Conclusões

Neste artigo abordou-se o desenvolvimento de um protótipo, com vista a estudar a adequadibilidade da tecnologia *SVG* à visualização de sinópticos na *Web*.

No que diz respeito à capacidade da tecnologia *SVG* para a representação e visualização de diagramas, interacção com o sistema *SCADA/DMS* e desempenho no carregamento dos gráficos *SVG*, os resultados mostraram-se satisfatórios. Por outro lado, quanto à capacidade de actualização dos diagramas em tempo real, de acordo com os eventos que recebe do sistema, os resultados não atingiram o mínimo aceitável. Este facto explica-se pela utilização da API DOM, o que implica uma elevada ocupação de memória e uma baixa performance no tratamento da árvore DOM (associada aos gráficos *SVG*).

Estes testes também demonstraram que esta baixa performance se deve à forma como os símbolos são actualizados no gráfico, ou seja, se a animação se basear na substituição dos símbolos (substituição de nós na árvore DOM) verifica-se uma baixa performance, por outro lado se esta se basear na alteração de propriedades dos símbolos (não há substituição de nós) a performance torna-se bastante elevada.

Em conclusão, os *SVG* mostram-se capazes de suportar a visualização dos diagramas, a comunicação com o sistema (GENESys) e também a interacção com o utilizador.

Como trabalho futuro fica a melhoria na implementação da animação dos diagramas, de acordo com a informação recebida do sistema. Este trabalho de-

verá passar pela alteração da representação dos símbolos em XML, por forma a permitir a animação através de mudanças apenas nos valores de propriedades.

Referências

[Apa05a] Apache. Batik *SVG* Toolkit Architecture. `http://xml.apache.org/batik/architecture.html#coreComponents`, Junho 2005.

[Apa05b] Apache. Batik SVG Toolkit: Scripting. `http://xml.apache.org/batik/scripting.html`, Maio 2005.

[Apa05c] Apache. Batik *SVG* Toolkit: *SVG* Generator. `http://xml.apache.org/batik/svggen.html/`, Junho 2005.

[Fer03] Ricardo Jorge Nogueira Fernandes. Interfaces Web para aplicações SCADA. Mestrado em Ciências de Computadores, Faculdade de Ciências da Universidade do Porto, Fevereiro 2003.

[Gar05] Rodrigo García García. *SVG* for SCADA. Swiss Federal Institute of Technology Lausanne (EPFL). `http://www.svgopen.org/2004/papers/SVGforSCADA/`, Abril 2005.

[Gom03] Miguel Ferreira Pereira Gomes. Canal SCADA na *Web*. Mestrado em Ciências de Computadores, Faculdade de Ciências da Universidade do Porto, Fevereiro 2003.

[IBM05] IBM. Program with SVG. `http://www-128.ibm.com/developerworks/xml/library/x-matters40/`, Maio 2005.

[LFCJ02] Simon Lok, Steven K. Freiner, William M. Chiong, and Yoav J.Hirsch. A graphical user interface toolkit approach to thin-client computing. *Proceedings of the Eleventh International Conference on World Wide Web*, pages 718–725, 2002.

[LSFH00] Sheng Feng Li, Quentin Stafford-Fraser, and Andy Hopper. Integrating synchronous and asynchronous collaboration with virtual networking computing. *Proceedings of the First International Workshop on Intellingent Multimedia Computing and Networking, Atlantic City, USA*, Vol. 2, pages 717–721, Fevereiro–Março 2000.

[Mar05a] Filipe Marinho. *Actividade Sem Concepção — Sinópticos na Web*. Relatório interno, *EFACEC Sistemas de Electrónica S.A.*, Maio 2005.

[Mar05b] Filipe Marinho. *Testes de Sistema — Sinópticos na Web*. Relatório interno, *EFACEC Sistemas de Electrónica S.A.*, Agosto 2005.

[Mic04] Microsoft. Windows server 2003 terminal services. `http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technolo%gies/featured/termserv/default.mspx`, 2004.

[MV01] Dave Marsh and Paulo Viegas. The bus architecture. *IEEE Porto PowerTech 2001* `http://power.inescn.pt/powertech/`, Setembro 2001.

[SLN99] Brian K. Schmidt, Monica S. Lan, and J.Duane Northcutt. The interactive performance of SLIM a stateless, thin-client architecture. *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles (SOSP)*, Vol. 34, pages 32–47, Dezembro 1999.

[Wor05a] W3C World Wide Web Consortium. W3C — About SVG. `http://www.w3.org/TR/SVG/intro.html/`, Abril 2005.

[Wor05b] W3C World Wide Web Consortium. W3C SVG Specification. `http://www.w3.org/TR/SVG11/`, Junho 2005.

[ZPMD97] Debora J. Zukowski, Apratim Purakayastha, Ajay Mohindra, and Murthy Devarakonda. Metis: A thin-client application framework. *Proceedings of the Third Conference on Object-Oriented Technologies and Systems*, pages 103–114, Junho 1997.