

SVG WebSlide & SVG WebChart : aplicações com gráficos vectoriais escaláveis em XML

Helder Filipe P.C. Ferreira

Faculdade de Engenharia Universidade do Porto
FEUP / Portugal
hfilipe@fe.up.pt

Abstracto

A maior parte das linguagens XML apenas exprime informação textual, com deficientes ou inexistentes capacidades de representação gráfica. Por esse motivo a W3C [1] desenvolveu o SVG [2] (*Scalar Vector Graphics*), e tornou “gráfico” o XML. As potencialidades gráficas, a expansibilidade e flexibilidade inerente a qualquer linguagem XML, e a possibilidade de publicação directa na web, fizeram do SVG a linguagem ideal para criação e publicação de conteúdos com qualidade gráfica excelente e imutável, independente do tamanho da janela do *browser* ou da resolução do monitor.

Este artigo relata os resultados de um trabalho em desenvolvimento [3], que visou a criação de duas aplicações práticas de SVG: para *criação automatizada de apresentações ou slides on-line (SVG WebSlide)*, e para a *criação automatizada de relatórios on-line com gráficos de dados estatísticos (SVG WebChart)*.

Ambas as aplicações motivaram a criação de duas novas linguagens XML: **Presentation Markup Language** como *XML para slides* de apresentações; e **Workbook Markup Language** como *XML para gráficos de dados*.

Palavras-chave: XML, SVG, slides, gráficos estatísticos, PresentationML, WorkbookML .

1 Introdução

SVG significa **Scalable Vector Graphics** (Gráficos Vectoriais Escaláveis):

- **Gráficos:** A maioria das linguagens XML apenas exprime informação textual, com deficientes ou inexistentes, capacidades de representação gráfica. Por esse motivo a W3C desenvolveu o SVG, e tornou “gráfico” o XML.
- **Vectoriais:** Actualmente existem dois tipos de gráficos: *raster* e *vectorial*. Os gráficos do tipo raster baseiam-se nos pixels para representar a imagem, enquanto que os gráficos vectoriais são compostos por polígonos, linhas, pontos e curvas. Estes últimos podem também incluir imagens do tipo raster. Uma das grandes vantagens dos gráficos vectoriais é o facto de estes não sofrerem efeito de *aliasing*.
- **Escaláveis:** Escalamento é o acto de aumentar ou diminuir de forma uniforme. Em termos gráficos isto significa que um objecto SVG não se encontra limitado ao tamanho fixo de um pixel. Isto é, podemos aumentar ou diminuir uniformemente um objecto SVG sem que este

perca definição gráfica. Uma das outras vantagens do SVG é o facto de podermos reutilizar objectos SVG dentro de um novo objecto SVG, escalando-os.

O conjunto destes três conceitos define as capacidades do SVG e justifica a sua criação como a tecnologia de gráficos para a *World Wide Web*.

A aplicação directa do SVG é a representação gráfica de algo. No entanto, desta representação podem surgir diversas ideias, tais como:

- Desenhos e animações em páginas web;
- Exemplos interactivos educativos;
- Slides de apresentações;
- Pesquisa virtual de edifícios e interiores de andares;
- Mapas geográficos;
- Representação gráfica de tabelas de dados estatísticos;
- Entre outros...

O objectivo deste trabalho é uma proposta de aplicações simples em SVG, com algumas sugestões ou propostas de linguagens de marcação para determinadas áreas de aplicação. Foram desenvolvidas duas aplicações práticas do SVG, visando:

- a *criação automatizada de slides* de apresentações (**SVG WebSlide**)
- e a *criação automatizada de representações gráficas de tabelas unidimensionais de dados estatísticos* (**SVG WebChart**).

Para ambas as aplicações foram desenvolvidas novas linguagens XML que compõem documentos que são processados via XSLT, usando o processador Saxon [4]. Toda a documentação, aplicações SVG e demonstrações on-line, encontram-se disponíveis em: <http://lpf-esi.fe.up.pt/~ped>.

2 SVG WebSlide: criação de slides

Uma das aplicações práticas que surgiu muito naturalmente, foi a elaboração de slides em SVG. As potencialidades gráficas e a possibilidade de publicação directa na Internet, são duas características que criam condições fantásticas para a construção de slides e a sua publicação imediata na web, com características de imagens vectoriais. Assim nasceu o **SVG WebSlide**.

Com esta aplicação pretende-se automatizar o processo de criação de uma apresentação dentro de moldes personalizáveis por parte dos utilizadores, através do uso de CSS [5], em que o primeiro e último slide da apresentação são gerados automaticamente, assim como um sistema de navegação entre slides.

2.1 Presentation Markup Language

A criação de slides em SVG, usando a própria linguagem do SVG, é um pouco entediante e complexa, obrigando o utilizador a ter conhecimentos algo profundos sobre determinados objectos SVG, propriedades e definições. Sendo assim, e para facilitar a construção de slides, optou-se pela criação de uma nova linguagem XML mais simples e intuitiva.

Foi realizada uma pesquisa na Internet, procurando saber se já existia uma linguagem XML própria para slides. De facto existe e chama-se **SlideML** [6]. No entanto, esta linguagem encontra-se ainda em desenvolvimento (não existe sequer uma DTD), e a sua definição é demasiado

complexa para a aplicação simples que se procurava implementar. Sendo assim optou-se por criar uma nova linguagem XML para slides, a que se deu o nome de **PresentationML**. O seu conjunto de etiquetas de marcação consiste no seguinte:

| |
|--|
| <p>Elemento: <code>presentation</code> : Delimita um conjunto de slides. Atributos: <code>title</code>: título da apresentação. <code>subtitle</code>: subtítulo da apresentação. <code>author</code>: autor(es) da apresentação. <code>date</code>: data em que foi feita, ou vai ser apresentada. <code>organization</code>: instituição ou organização a que o(s) autor(es) pertence(m). <code>dir</code>: directório no qual irão ser criados os slides em SVG.</p> <p>Elemento: <code>slide</code> : Define um slide. Atributos: <code>subtitle</code>: título do slide.</p> <p>Elemento: <code>tline</code> : Linha de texto. Elemento: <code>blist</code> : Lista de itens. Elemento: <code>bitem</code> : Item de uma lista.</p> <p>Elemento: <code>link</code> : Endereço URL/URI. Atributos: <code>url</code>: endereço.</p> <p>Elemento: <code>image</code> : imagem do tipo JPG ou PNG ou SVG. Atributos: <code>x</code>: posição no eixo das abcissas. <code>y</code>: posição no eixo das ordenadas. <code>width</code>: largura da imagem. <code>height</code>: altura da imagem. <code>url</code>: endereço da imagem.</p> <p>Elemento: <code>code</code> : Excerto de um programa ou rotina de código. Elemento: <code>br</code> : Nova linha de texto.</p> <p>Elemento: <code>paint</code> : Pinta o texto com uma determinada cor. Atributos: <code>color</code>: cor em hexadecimal ou em rgb.</p> <p>Elemento: <code>space</code> : Espaçamento ou afastamento. Atributos: <code>n</code>: posição de afastamento em relação à margem esquerda.</p> <p>Elemento: <code>b</code> : Negrito. <code>i</code> : Itálico. <code>u</code> : Sublinhado.</p> |
|--|

Tabela 1: Conjunto de etiquetas de marcação *PresentationML*.

| |
|---|
| <pre><!-- PRESENTATION : elemento raiz do documento --> <!ELEMENT presentation (slide)+> <!ATTLIST presentation title CDATA #REQUIRED> <!ATTLIST presentation date CDATA #REQUIRED> <!ATTLIST presentation author CDATA #REQUIRED> <!ATTLIST presentation subtitle CDATA #IMPLIED> <!ATTLIST presentation organization CDATA #IMPLIED> <!ATTLIST presentation dir CDATA #IMPLIED></pre> |
|---|

```

<!-- SLIDE : slide -->
<!ELEMENT slide (tline | blist | bitem | link | image | code | br | paint | space | b | i | u)+>
  <!ATTLIST slide subtitle CDATA #REQUIRED>

<!-- TLINE : linha de texto -->
<!ELEMENT tline ( #PCDATA | link | code | br | paint | space | b | i | u)*>
<!-- BLIST : lista de itens -->
<!ELEMENT blist (bitem)+>
<!-- BITEM : item de uma lista -->
<!ELEMENT bitem ( #PCDATA | link | code | br | paint | space | b | i | u)*>

<!-- LINK : endereço URI / URL -->
<!ELEMENT link ( #PCDATA)>
  <!ATTLIST link url CDATA #REQUIRED>

<!-- IMAGE : imagem -->
<!ELEMENT image EMPTY>
  <!ATTLIST image x CDATA #REQUIRED>
  <!ATTLIST image y CDATA #REQUIRED>
  <!ATTLIST image width CDATA #REQUIRED>
  <!ATTLIST image height CDATA #REQUIRED>
  <!ATTLIST image url CDATA #REQUIRED>

<!-- CODE : rotinas de código -->
<!ELEMENT code ( #PCDATA | br | paint | space | b | i | u)*>
<!-- BR : nova linha de texto -->
<!ELEMENT br EMPTY>

<!-- PAINT : pinta texto com uma determinada cor -->
<!ELEMENT paint ( #PCDATA | link | code | br | paint | space | b | i | u)*>
  <!ATTLIST paint color CDATA #REQUIRED>

<!-- SPACE : espaçamento -->
<!ELEMENT space EMPTY>
  <!ATTLIST space n CDATA #REQUIRED>

<!-- B : negrito -->
<!ELEMENT b ( #PCDATA | link | code | br | paint | space | i | u)*>
<!-- I : itálico -->
<!ELEMENT i ( #PCDATA | link | code | br | paint | space | b | u)*>
<!-- U : sublinhado -->
<!ELEMENT u ( #PCDATA | link | code | br | paint | space | b | i)*>

```

Tabela 2: *Document Type Definition do PresentationML.*

2.2 Conversão do PresentationML em SVG

A transformação de PresentationML para SVG é feita através de uma stylesheet (**WebSlide.XSL**). Esta funciona recorrendo a chamadas de *templates* pré-definidos, ou aplicando templates para os elementos identificados. Quando um elemento de PresentationML é detectado, este é substituído por um conjunto de marcações SVG. O utilizador não necessita de se preocupar com o slide de rosto e o último de agradecimento, uma vez que estes são gerados automaticamente. No processo de conversão existem 3 aspectos interessantes que merecem ser mencionados:

- a necessidade de se criarem vários documentos SVG (um para cada slide);
- um mecanismo de navegação entre os slides;
- posicionamento dos elementos gráficos no slide.

Para a criação de vários documentos, para cada slide é definido um novo ficheiro de saída dentro da pasta onde estes são guardados, com o nome "slide"+posição do nó slide+".svg". Deste modo, são sequencialmente criados ficheiros slideX.svg.

Para o mecanismo de navegação foram definidas algumas variáveis que capturavam as posições dos nós slide. Uma vez que o nome dos slides se baseava na posição, seria apenas necessário adicionar 1 à posição corrente para avançar, e subtrair 1 para recuar na apresentação.

O posicionamento dos elementos gráficos é realizado de modo diferente consoante o elemento em causa é texto ou uma imagem/forma geométrica. O texto em SVG utiliza atributos x , y , dx e dy que permitem posicionar o texto no slide de modo absoluto ou em relação ao último posicionamento de texto. As imagens ou formas geométricas são posicionadas de modo absoluto usando os atributos x e y .

A versão integral desta stylesheet pode ser vista em: <http://lpf-esi.fe.up.pt/~ped/down.html> .

2.3 Exemplos de apresentações criadas pelo SVG WebSlide

Vejamos um exemplo de uma apresentação com um único slide em PresentationML:

```
<presentation title="Exemplo do WebSlide" author="Helder Ferreira" date="13-Nov-2003" organization="PED / MEI / FEUP">
  <slide subtitle="Slide criado por este programa">
    <tline>Exemplo do <paint color="navy"><b>SVG</b></paint> <i><paint color="orange">WebSlide</paint></i></tline>
    <blist>
      <bitem>É um aplicativo desenvolvido em <b>XHTML+CSS+XSL+<paint color="red">SVG</paint>+PERL</b>.</bitem>
    </blist>
    <tline>Mais info: <link url="http://www.fe.up.pt/~hfilipe">http://lpf-esi.fe.up.pt/~ped</link></tline>
  </slide>
</presentation>
```

Tabela 3: Exemplo de um documento PresentationML.

A figura seguinte mostra o slide que foi gerado a partir do documento da tabela 3.



Figura 1: Slide gerado pelo SVG WebSlide.

Outros exemplos podem ser vistos na página web do SVG WebSlide, ou mesmo testados, usando a demonstração on-line em: <http://lpf-esi.fe.up.pt/~ped> .

3 SVG WebChart: criação de gráficos de tabelas de dados

O SVG WebChart é uma aplicação cujo objectivo é a geração automatizada de gráficos em SVG, a partir de tabelas de dados, definidos através de uma linguagem XML (WorkbookML), baseada no XML do Microsoft Excel. Esta aplicação apenas suporta tabelas de dados no formato *{descrição, dado}*. Isto é, tabelas do tipo:

| Venda de Livros num ano | |
|-------------------------|-----------|
| Categoria | Valor |
| Policia | 1.051.234 |
| Romance | 2.345.678 |
| Poesia | 345.893 |
| Escolar | 5.235.623 |

Tabela 4: Exemplo de tabela passível de ser representada pelo SVG WebChart.

Isto deve-se ao facto de ser necessária uma grande complexidade nos cálculos matemáticos que calculam o posicionamento dos elementos gráficos, legendas, etc. Uma vez que o objectivo do trabalho era apenas concretizar representações simples de gráficos estatísticos, optou-se pelo uso de tabelas com apenas uma série de valores.

O WebChart suporta diferentes tipos de gráficos: **Barras**, **Pontos** ou **Linhas**. E suporta diferentes tipos de direcções dos gráficos: **Vertical** (0° de orientação) ou **Horizontal** (90° de orientação).

Nota: Assume-se que a posição natural de um gráfico de dados é a posição vertical.

3.1 Workbook Markup Language

A criação de gráficos em SVG, usando a própria linguagem do SVG, é algo complexa porque exige um elevado número de cálculos, escalamentos, translações e rotações de objectos SVG.

Para não obrigar o utilizador a ter conhecimentos profundos sobre determinados objectos SVG, propriedades e definições, criou-se o SVG WebChart, e uma linguagem XML mais simples e intuitiva.

Uma vez que a folha de cálculo actualmente mais usada é o *Microsoft Excel*, optou-se por estudar o ficheiro XML gerado pelo mesmo quando se cria uma tabela de dados. No entanto, o documento XML gerado pelo Excel é mais vocacionado para descrever visualmente a área de trabalho da aplicação e por isso contém demasiadas etiquetas de marcação desnecessárias, tendo em conta o que se pretendia neste trabalho. Sendo assim, optou-se por criar uma nova linguagem XML para gráficos, baseada no XML do Microsoft Excel, a que se deu o nome de **WorkbookML**. O seu conjunto de etiquetas de marcação consiste no seguinte:

```
Elemento: Workbook : Delimita um conjunto de gráficos.
Atributos:


- name: título do conjunto de gráficos.
- subtitle: subtítulo do conjunto de gráficos.
- logo: eventual logotipo de um relatório estatístico.
- logow: largura do logotipo.
- logoh: altura do logotipo.
- dir: directório no qual irão ser criados os gráficos em SVG.

```

Elemento: `DocumentProperties` : Define meta-informação do documento de gráficos.
 Elemento: `Author` : Autor do documento.
 Elemento: `Created` : Data/Hora de criação do documento.
 Elemento: `Company` : Instituição/Organização à qual o autor pertence.
 Elemento: `Version` : Versão do documento.

Elemento: `Worksheet` : Definição de uma folha de dados estatísticos.
 Atributos:

- `name`: título do gráfico de dados.
- `orientation`: orientação do gráfico (vertical - 0°, horizontal - 90°).
- `type`: tipo de gráfico: (barras - bars, pontos - points e linhas - lines).

Elemento: `Table` : Define um gráfico/tabela de dados.
 Elemento: `Caption` : Legendas dos eixos do gráfico.
 Elemento: `Row` : Linha da tabela de dados.
 Elemento: `Cell` : Coluna da tabela de dados.

Elemento: `Data` : Dados da tabela.
 Atributos:

- `type`: tipo de dado (*string* ou *number*).

Tabela 5: Conjunto de etiquetas de marcação *WorkbookML*.

```

<!-- WORKBOOK : elemento raiz do documento -->
<!ELEMENT Workbook (DocumentProperties, Worksheet+)>
  <!ATTLIST Workbook name CDATA #REQUIRED>
  <!ATTLIST Workbook subtitle CDATA #REQUIRED>
  <!ATTLIST Workbook logo CDATA #IMPLIED>
  <!ATTLIST Workbook logow CDATA #IMPLIED>
  <!ATTLIST Workbook logoh CDATA #IMPLIED>
  <!ATTLIST Workbook dir CDATA #IMPLIED>

<!-- DOCUMENTPROPERTIES : meta-informação do documento -->
<!ELEMENT DocumentProperties (Author, Created, Company, Version)>
<!-- AUTHOR : autor do documento -->
<!ELEMENT Author (#PCDATA)>
<!-- CREATED : data/hora da criação do documento -->
<!ELEMENT Created (#PCDATA)>
<!-- COMPANY : instituição ou organização à qual o autor pertence -->
<!ELEMENT Company (#PCDATA)>
<!-- VERSION : versão do documento -->
<!ELEMENT Version (#PCDATA)>

<!-- WORKSHEET : folha de dados estatísticos -->
<!ELEMENT Worksheet (Table)>
  <!ATTLIST Worksheet name CDATA #REQUIRED>
  <!ATTLIST Worksheet orientation (0 | 90) #REQUIRED>
  <!ATTLIST Worksheet type (bars | lines | points) #REQUIRED>

<!-- TABLE : tabela de dados -->
<!ELEMENT Table (Caption, Row+)>
<!-- CAPTION : legenda dos eixos -->
<!ELEMENT Caption (Cell)+>
<!-- ROW : linha da tabela de dados -->
<!ELEMENT Row (Cell)+>
<!-- CELL : coluna da tabela de dados -->
<!ELEMENT Cell (Data)>
  
```

```

<!-- DATA : dados -->
<!ELEMENT Data (#PCDATA)>
  <!ATTLIST Data type (Number | String) #REQUIRED>

```

Tabela 6: Document Type Definition do WorkbookML.

3.2 Conversão de WorkbookML em SVG

A transformação de WorkbookML para SVG é feita através de uma stylesheet (**WebChart.XSL**). Uma vez que o sistema de coordenadas do SVG é complexo, assim como os cálculos necessários para construir o gráfico, optou-se por desenhá-lo na horizontal. Posteriormente, aplicam-se operações de translação e rotação para colocar o gráfico desenhado no local pretendido (centro do slide). Seja qual for a orientação e o tipo de gráfico, é sempre necessário determinar:

- a **largura de cada barra** que forma o gráfico (mesmo nos gráficos de pontos, pois estes também são formados por retângulos, mas com uma área menor).
- a **barra com valor mais elevado** (para que se possam escalar todos os objectos SVG, principalmente os eixos e o texto).;

A versão integral desta stylesheet pode ser vista em: <http://lpf-esi.fe.up.pt/~ped/down.html> .

3.3 Exemplos de apresentações criadas pelo SVG WebChart

Um exemplo de um documento WorkbookML bem-formatado e válido seria:

```

<Workbook name="Relatório Anual do Restaurante «Tripas à Moda do Porto»" subtitle="Ano 2003">
  <DocumentProperties>
    <Author>Helder Filipe P.C. Ferreira</Author>
    <Created>27-12-2003</Created>
    <Company>PED / MEI / FEUP</Company>
    <Version>1.0</Version>
  </DocumentProperties>
  <Worksheet name="Pratos de Carne do Restaurante «Tripas à Moda do Porto»" orientation="0" type="bars">
    <Table>
      <Caption>
        <Cell><Data type="String">Pratos de Carne</Data></Cell>
        <Cell><Data type="String">Nº de refeições servidas</Data></Cell>
      </Caption>
      <Row>
        <Cell><Data type="String">Vitela Assada</Data></Cell>
        <Cell><Data type="Number">10456</Data></Cell>
      </Row>
      <Row>
        <Cell><Data type="String">Tripas à moda do Porto</Data></Cell>
        <Cell><Data type="Number">98020</Data></Cell>
      </Row>
      <Row>
        <Cell><Data type="String">Cozido à Portuguesa</Data></Cell>
        <Cell><Data type="Number">56000</Data></Cell>
      </Row>
      <Row>
        <Cell><Data type="String">Lombo Assado</Data></Cell>
        <Cell><Data type="Number">12345</Data></Cell>
      </Row>
    </Table>
  </Worksheet>
</Workbook>

```

```

<Cell><Data type="String">Prego no Prato</Data></Cell>
<Cell><Data type="Number">34023</Data></Cell>
</Row>
</Table>
</Worksheet>
</Workbook>

```

Tabela 7: Exemplo de documento em WorkbookML.

A figura seguinte mostra o gráfico que foi gerado a partir do documento da tabela 7.

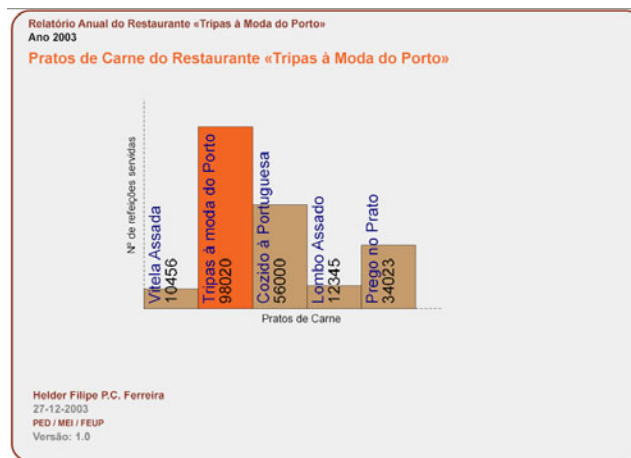


Figura 2: Gráfico gerado pelo SVG WebChart.

Outros exemplos podem ser vistos na página web do SVG WebChart, ou mesmo testados, usando a demonstração on-line em: <http://lpf-esi.fe.up.pt/~ped>.

4 Modo de funcionamento das aplicações

Foram desenvolvidos dois programas de linha de comando, que realizam todas as operações necessárias à transformação de PresentationML ou WorkbookML em SVG. Estes programas foram desenvolvidos em Perl [7], e recorrem ao processador XSLT, Saxon. Para se usar o Saxon é necessário possuir o Microsoft Java Virtual Machine [8] instalado. Para que se possam visualizar os slides criados, é recomendado o plugin da Adobe - *SVG Viewer*, e o uso do *browser Internet Explorer*. As sintaxes de comando são da forma: `webslide.exe exemplo.xml` e `webchart.exe -help`. Cada um dos programas realiza a seguinte sequência de operações:

- Leitura do documento PresentationML ou WorkbookML.
- Testa a presença do elemento raiz: Presentation ou Workbook.
- Usa o módulo de Perl, XML::Parser [9] para validar o documento XML.
- Pesquisa pelo atributo `dir` do elemento Presentation ou Workbook, de modo a descobrir qual a pasta onde os slides devem ser criados. Se o atributo não tiver sido colocado ou estiver em branco, ele assume por defeito que a pasta denomina-se “*slides*”, no caso do WebSlide, ou “*charts*”, no caso do WebChart.

- Cria a pasta para guardar os ficheiros criados.
- Modifica todos os endereços para ficheiros externos, de modo a que estes incluam a referência para o directório onde vão ser criados os slides.
- Copia o ficheiro CSS e imagens que são usadas na apresentação para esse directório.
- Utiliza o Saxon para realizar a transformação com o WebSlide.XSL ou o Webchart.XSL [10].
- Cria os slides e retorna sucesso ao autor.

5 Conclusões e Perspectivas Futuras

O desenvolvimento destas duas aplicações veio:

- Facilitar a criação de apresentações on-line, com uma qualidade vectorial, e por essa razão legíveis e acessíveis em qualquer resolução ou tamanho;
- Simplificar a criação de apresentações em SVG e permitir a reutilização de slides entre apresentações, uma vez que basta um simples *copy & paste* de uma apresentação para outra.
- Facilitar a criação de gráficos de dados estatísticos, embora ainda limitados na dimensionalidade das tabelas.
- Estabelecer uma proposta de linguagens de marcação XML para a criação de slides e de gráficos, que até ao momento, ou eram inexistentes ou demasiado complexas.
- Fornecer meios de transformação SVG de distribuição gratuita.

Actualmente estas aplicações encontram-se na versão 1.0, e com algumas limitações, principalmente o SVG WebChart. Planeia-se no futuro, melhorar as capacidades de ambas as aplicações. Para o SVG WebSlide, prevê-se um melhoramento na quantidade de objectos a suportar, assim como um aumento no grau de personalização dos slides. Para o SVG WebChart, prevê-se um aumento do tipo de gráficos a suportar, assim como o suporte para tabelas de dados multidimensionais.

Referências

- [1] World Wide Web Consortium (W3C) - <http://www.w3c.org> .
- [2] Scalable Vector Graphics (SVG) - <http://www.w3.org/Graphics/SVG> .
- [3] SVG WebSlide e SVG WebChart - <http://lpf-esi.fe.up.pt/~ped> .
- [4] Saxon - <http://saxon.soundforge.net/> .
- [5] Cascading Style Sheet (CSS) - <http://www.w3.org/Style/CSS/> .
- [6] SlideML - <http://www.slideml.org> .
- [7] Practical Extraction and Report Language (Perl) - <http://www.perl.org> e <http://www.perl.com> .
- [8] Microsoft Java Virtual Machine - <http://java-virtual-machine.net/download.html> .
- [9] XML::Parser - <http://search.cpan.org/~msergeant/XML-Parser-2.34/Parser.pm> .
- [10] Stylesheets que transformam PresentationML ou WorkbookML em SVG - <http://lpf-esi.fe.up.pt/~ped/down.html> .