La dualidad texto-imagen en *SVG* (*scalable vector graphics*): nuevas posibilidades para la descripción de información gráfica

Artículo

Por **Antonio de la Rosa** y **Jose A. Senso**

Antonio de la Rosa es licenciado en Documentación por la Universidad de Granada. Trabaja desde hace cuatro años en la empresa holandesa Wisdom Information Consultants como Senior Consultant. Sus líneas de investigación están relacionadas con la gestión del conocimiento y de la información estructurada. Está especializado en XML, Topic Maps y XTM.

Resumen: *La web es un fenómeno multimedia en cuanto a contenidos y presentación. La cantidad de contenido gráfico y/o multimedia es comparable a la cantidad de contenido textual. Sin embargo, cuando se trata de describir y recuperar información, el texto parece ser el único recurso. La dualidad texto-imagen de SVG (scalable vector graphics) plantea otras posibilidades.*

Palabras clave: *SVG (scalable vector graphics), Recuperación de información multimedia, Visualización de la información.*

Title: The text-image duality in *SVG* (*scalable vector graphics*): new possibilities for the description of graphic information

Abstract: *The web is a multimedia phenomenon. The amount of online graphical content is comparable to the amount of textual content. Yet, when it comes to describing and retrieving information, text seems to provide the only means. The text-image duality of SVG (scalable vector graphics) raises other possibilities.*

Keywords: *SVG (scalable vector graphics), Multimedia information retrieval, Information visualisation.*

***Rosa, Antonio de la** y **Senso, Jose A**. "La dualidad texto-imagen en SVG (Scalable Vector Graphics): nuevas posibilidades para la descripción de información gráfica". En: El profesional de la información, 2003, septiembre-octubre, v. 12, n. 5, pp. 377-398.*

Jose A. Senso, doctor en Documentación, es profesor de la Universidad de Granada desde 1998 y subdirector de la revista El profesional de la información. Sus líneas de investigación se centran en sistemas de metadatos, especialmente rdf, el desarrollo e implementación de la web semántica y Topic Maps.

1. Marco y finalidad de este artículo

El éxito de la web se debe, en gran medida, al hecho de haber popularizado el contenido multimedia dejando atrás lo exclusivamente textual. Sin embargo, el usuario no reacciona de la misma forma ante un documento y una fotografía. Generalmente, se sigue identificando información con contenido textual.

Esto ocurre especialmente en el área de la recuperación de información (en adelante RI) digital. La implementación de la RI en la web dista mucho de ser ideal, pero se puede afirmar que existen mecanismos eficaces para extraer los conceptos presentes en el texto de un documento y medir la relevancia del mismo como respuesta a una consulta.

Esencialmente se asume que los temas de los que trata un documento están presentes en el texto como conceptos que pueden formalizarse mediante descriptores. En este sentido, la gran ventaja de la RI textual es que la forma de consultar una colección de documentos es la misma que la de describirlos: en ambos casos se utiliza un conjunto de palabras.

La recuperación de imágenes es más compleja. El concepto central de una imagen es más difícil de extraer puesto que una imagen puede interpretarse de más formas que un texto. Si tomamos por ejemplo una fotografía de la Sagrada Familia, podríamos describirla de diferentes formas: una imagen religiosa, una imagen arquitectónica, una imagen en la que lo vertical predomina sobre lo horizontal y las formas curvas sobre las rectas, un icono turístico de Barcelona, etcétera. El problema es saber cuál de estas descripciones es la más apropiada y qué características de la imagen van a usar los usuarios al formalizar una consulta para intentar recuperarla.

Artículo recibido el 04-03-03
Aceptación definitiva: 14-07-03

El hecho es que, con frecuencia, los usuarios buscan imágenes y otros media en la web. Existen, por ejemplo, numerosos repositorios de imágenes fotográficas que han debido implementar sistemas de búsqueda. Éstos se han basado en características tales como el nombre del fichero, la categoría dentro de la cual se almacenan, etcétera. Entre esos repositorios destacan el catálogo *Corbis* de *Microsoft*[2], el archivo de noticias audiovisuales de la *CNN*[3], la *Lockheed Martin digital photo collection*[4], el archivo de fotografías de la *Nasa*[5] o el *Department of Energy digital archive*[6].

¿Cómo se busca una imagen en una colección de millones de imágenes? A grosso modo ese proceso se puede realizar de tres formas diferentes:

1.- El usuario trata de recuperar una imagen particular que conoce y que sabe que está presente en la colección. En este caso es posible utilizar en la recuperación de la imagen sus características "primarias": forma, color, dimensiones, textura, etcétera. Este tipo de consulta "visual" suele ser un añadido de la textual y se implementa con más frecuencia sobre dominios restringidos como la medicina, geología, química, etc., donde es más sencillo formalizar un modelo de conocimiento coherente en el que se integre mejor cualquier tipo de información, bien sea gráfica o textual.

Un ejemplo de sistema de búsqueda que implementa la consulta visual es *VisualSeek*[7], que integra el modelo de búsqueda espacial empleado en los sistemas de información geográfica y el de consulta visual. Los usuarios pueden pedir al sistema que localice imágenes con características visuales y/o espaciales similares a las especificadas en la consulta. La figura 1 muestra una sesión en la que la consulta consiste en dos áreas de color (azul y verde) localizadas en posiciones espaciales determinadas. Esta consulta sirve para localizar imágenes con un patrón de cielos azules y campos verdes.

Otros proyectos desarrollados en la misma línea son:

—*Blobworld*:
http://elib.cs.berkeley.edu/photos/blobworld/

—*Cbir (Content-based image retrieval)*:
http://www-db.stanford.edu/IMAGE/

—*Oversized color images*:
http://www.columbia.edu/dlc/nysmb/

—*Qbic (Query by image content)*:
http://wwwqbic.almaden.ibm.com/

—*Squid (Shape queries using image databases)*:
http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.htm

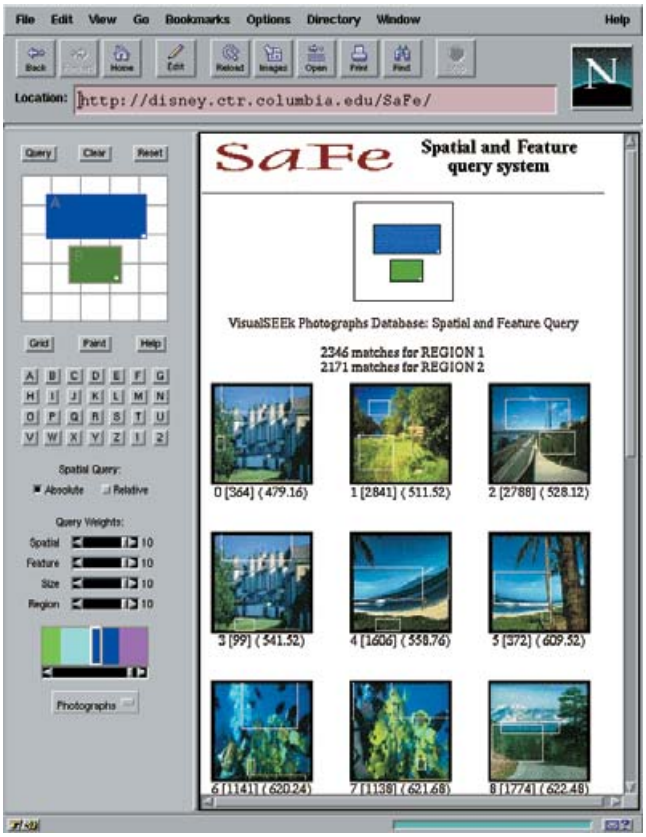


Figura 1. Fuente:
http://www.dlib.org/dlib/february97/columbia/02chang.html

2.- El usuario "navega" dentro de la colección en busca de imágenes desconocidas. En este caso el proceso es comparable al de la RI textual. En el modelo textual de recuperación se construye un índice[8] partiendo de un conjunto de documentos contra el cual los usuarios lanzan consultas. Éstas se gestionan muchas veces empleando las mismas técnicas utilizadas para la indización, aunque algunas veces se requiera un proceso más especializado, por ejemplo, la expansión de consultas usando un tesauro. Al final, la recuperación de información sigue siendo un procedimiento basado en la comparación de consultas con un índice.

La consulta que se formula trata de definir los tópicos relacionados con la imagen que se busca. Esa consulta se lanza contra un índice textual. La correspondencia entre ese índice y la colección de imágenes que describe es generalmente mucho más pobre que la que existe entre un índice textual y una colección de documentos textuales. Lógicamente, también los resultados de la búsqueda serán mucho más pobres.

3.- El usuario introduce una serie de términos de búsqueda dentro de determinados buscadores. Éstos realizan la consulta sobre los índices generados a partir de los atributos que rodean a cualquier imagen en Internet: su nombre, el texto adyacente a ésta, tamaño, etc. En algunas ocasiones este proceso se depura con filtros que se encargan de eliminar duplicaciones, pre-

cisar la calidad de la imagen, etc. Es el utilizado por *Google*, *AltaVista* o *AlltheWeb*.
http://images.google.com
http://images.av.com
http://www.alltheweb.com

En definitiva, parece necesario hallar un medio neutral en el que sea posible identificar las características del contenido de los objetos gráficos, describirlas con la misma precisión con la que en la actualidad es posible hacerlo con contenidos textuales y elaborar índices a partir de ellas. En el presente artículo se analizarán las posibilidades de *SVG*[1] en este sentido.

2. Información gráfica en la web

2.1. Tipos de contenido gráfico

Esencialmente existen dos tipos de contenido gráfico en la web:

Imágenes[9]: normalmente se trata de la versión digitalizada de imágenes ya existentes como fotografías, etc., aunque a veces son creaciones digitales realizadas con editores de imágenes como *Paint Shop Pro*, *Adobe Photoshop,* etc. Físicamente son mapas de bits: representaciones de imágenes almacenadas en la memoria del ordenador, compuestas por filas y columnas de puntos. El valor de cada punto se expresa mediante bits: para imágenes monocromas basta un bit por punto, para imágenes en color o tonalidades de gris se requieren más bits por punto. Cuantos más bits, más colores y tonos de gris podrán representarse. Si, por ejemplo, queremos representar los colores rojo, amarillo y negro, en el sistema *RGB* (*red, green, blue*), el rojo tendría la codificación 255,0,0, el amarillo 0,255,0 y el negro 0,0,0. El mapa de bits resultante sería (255,0,0),(0,255,0),(0,0,0).

La densidad de los puntos o resolución determina el nivel de detalle con el que va a definirse la imagen. La resolución se expresa normalmente en puntos por pulgada o *dpi* (*dots per inch*), o, también, mediante el numero de filas y columnas, como por ejemplo 640 por 480. De los diferentes métodos de almacenamiento y compresión de los mapas de bits resultan los diferentes formatos:

—*ACR*

—*BMP (Windows bitmap format)*

—*GIF (Graphics interchange format)*

—*PNG (Portable network graphics)*

—*PCX*

—*TGA (Targa)*

—*Tiff (Tagged Image File Format)*

Gráficos: generalmente son imágenes producidas por un ordenador y almacenadas físicamente como un conjunto de vectores y operaciones geométricas asociadas a ellos. Cada segmento de un gráfico se representa como un vector, definido por sus dos puntos extremos en una matriz x-y. La imagen constituye una lista de vectores denominada lista de presentación, a la que se añaden ecuaciones matemáticas que evitan la distorsión de los vectores durante las diferentes etapas de edición.

Los gráficos vectoriales tienen más flexibilidad que los mapas de bits, ya que presentan el mismo aspecto incluso después de que su tamaño haya sido aumentado o disminuido. Por el contrario, los mapas de bits son inutilizables una vez redimensionados. Además, las imágenes representadas mediante gráficos vectoriales ocupan menos memoria que las representadas mediante mapas de bits, de ahí que sea una tecnología muy apreciada en la web. Por otra parte, las imágenes almacenadas como vectores presentan una mejor apariencia en monitores e impresoras de alta resolución, mientras que los mapas de bits siempre se ven igual, independientemente de la resolución empleada. Casi todos los sistemas de gráficos más sofisticados, incluidos los sistemas *CAD* (*computer aided design*) y el software de animación, utilizan gráficos vectoriales. Alguno de los formatos más habituales:

—*CDR*

—*DRW*

—*DXF (drawing exchange format)*

—*EPS (encapsulated postscript)*

—*Flash*

—*GKS*

—*Pict*

—*Postscript*

—*PPT*

Una de las tecnologías basadas en gráficos vectoriales más popular en la web es el formato de animación gráfica *Shockwave Flash*. Con esta aplicación, de la empresa *Macromedia,* es posible crear animaciones o importar otras imágenes basadas en gráficos vectoriales. Estas animaciones o imágenes, conocidas como *Flash movies*, son independientes del navegador con el que se consultan. Esto significa que ni *Netscape* ni *Microsoft* han desarrollado sus propios parsers para *Flash* y tienen que usar un plugin de *Macromedia*.

2.2. Descripción de objetos gráficos

Una de las grandes diferencias entre imágenes y gráficos, en el contexto de la RI, es el nivel de profun-

didad con que se puede describir su contenido. Las imágenes básicamente contienen información que se refiere a la distribución de sus puntos: tamaño, resolución y, sobre todo, color. Su indización, por lo tanto, debe: o bien basarse en esos datos, o bien elaborar información sobre ellos mediante complicados procesos de reconocimiento de patrones, etcétera, o remitirse a un proceso de conceptualización textual por parte de un indizador humano. Los gráficos, sin embargo, contienen información muy detallada sobre su estructura y sobre la combinación de sus componentes. Una conclusión interesante a la que se retornará a lo largo del artículo es que los gráficos pueden descomponerse en sub-unidades descriptibles y las imágenes no.

Metadatos. Los gráficos, por lo tanto, se pueden describir mediante el uso de su información "primaria"[10]. Otro medio es la utilización de metadatos. En el marco de este artículo se entiende por metadatos toda aquella información descriptiva sobre las características y el contexto de un recurso gráfico digital y su contenido. Del mismo modo que existen sistemas de metadatos ideados para ser utilizados con información textual (*Dublin core metadata initiative*, *text encoding initiative*, *encoding archival description*, etc.) se han desarrollado otros para describir información multimedia:

—*CNI/Oclc image metadata workshop (DC3)*: *http://www.dlib.org/january97/oclc/01weibel.html*

—*DIG2000 initiative*: *http://www.digitalimaging.org/i_dig2000.html*

—*Niso/Clir/RLG technical metadata for images workshop*: *http://www.niso.org/imagerpt.html*

—Rdf *(resource description framework)*: *http://www.w3c.org/RDF*

—*Mpeg7 (Moving picture experts group)*: *http://mpeg.telecomitalialab.com/*

—*Vads (Visual arts data service)*: *http://vads.ahds.ac.uk/index.html*

—*VRA core: http://www.oberlin.edu/~art/vra/wc1.html*

Uno de los que más éxito ha tenido es *DC3*, producto de la tercera reunión de trabajo del equipo de la *Oclc* que desarrolló *Dublin core*. En esencia se trata de aplicar el modelo de 15 metaetiquetas *Dublin core* propuesto para recursos web a la descripción de imágenes de cualquier tipo. Otro sistema que se utiliza con mucha frecuencia es rdf. La diferencia entre rdf y *DC* es que el origen de rdf se encuentra en la descripción de documentos textuales de cualquier tipo, mientras que el segundo se aplica exclusivamente a objetos digitales. Ambos sistemas son especialmente adecuados para objetos digitales cuyas especificaciones se basen en la norma sgml, como es el caso de html, xml y *SVG*.

Mpeg7 es, desde julio de 2002, una norma *ISO/IEC* (*JTC1/SC29/WG11*) desarrollada por el *Mpeg* (*Moving picture experts group*) que ofrece un conjunto de herramientas para describir el contenido multimedia de imágenes estáticas, gráficos, modelos 3D, audio y vídeo, etc. El punto fuerte de este estándar está en que las descripciones pueden encontrarse físicamente asociadas al material audiovisual o en cualquier otra parte, señalando con un puntero al objeto fuente. Las imágenes se pueden describir con diferentes niveles de profundidad y utilizando diferentes niveles de abstracción. El principal problema que presenta este sistema es que para incluir determinados datos (gestión de derechos de autor, fechas de vida del objeto...) es necesario acudir a otros sistemas de metadatos, como *DC3*.

Junto a estos sistemas normalizados de metadatos existen otros que son propios de los formatos gráficos con los que se trabaja. Así por ejemplo, *PNG* (*portable network graphics*) cuenta con métodos para facilitar no sólo su descripción sino también su indización. *Jpeg* (*joint photographic experts group*) si bien no es exactamente un formato gráfico sino un algoritmo de compresión de imágenes, puede añadir segmentos en el propio fichero que desempeñen la función de marcador, describiendo los datos que van a identificar la representación visual del archivo. *WMF* (*Windows metafile*) permite añadir información relativa al contenido del gráfico, dimensiones, visualización recomendada o, incluso, llamadas a otros programas para que se ejecuten de forma automática al mismo tiempo que se abre el fichero.

Mapas de tópicos. *XTM* (*xml topic maps*) es uno de los formatos de metadatos más innovadores de la actualidad. Al utilizar este formato la descripción de un objeto digital se convierte en un mapa de tópicos. Los mapas de tópicos son, como en el caso de *SVG*, documentos sujetos a la sintaxis xml[11]. Un gráfico *SVG*, por ejemplo, es un documento xml que contiene elementos gráficos como un rectángulo o un círculo. En el caso de *XTM*, los elementos son abstracciones del contenido del objeto digital que describen, como un tópico o tema. Por ejemplo, el tópico central de este artículo es "*SVG*". La formalización de esos elementos y de las relaciones entre ellos da como resultado un mapa[12] del objeto en cuestión.

Digamos, por ejemplo, que queremos elaborar un mapa de tópicos sobre este artículo. Lo que obtendríamos sería un conjunto de elementos entre ellos:

—Tópicos: o temas de los que versa el artículo, por ejemplo *SVG*, xml, recuperación de información, etc.

—Tipos de tópicos: por ejemplo xml y *SVG* son ambos tópicos del tipo "lenguajes de etiquetas".

—Nombres de tópicos: *SVG* puede aparecer en el artículo como *SVG* o como *scalable vector graphics*.

—Asociaciones: una asociación de tópicos es un elemento vinculador que establece una relación entre dos o más tópicos. Por ejemplo: "xml es la sintaxis de *SVG*" o "*SVG* es el vocabulario gráfico de la sintaxis xml". Las asociaciones también pueden agruparse en tipos de asociaciones.

—Ocurrencias: un tópico puede ser vinculado a uno o más recursos de información considerados relevantes. A estos recursos se les llama ocurrencias. Una ocurrencia de *SVG* puede ser por ejemplo este documento, una sede web sobre *SVG*, una monografía, un comentario o cualquier otro tipo de recurso.

El código de ese mapa de tópicos, así como el tópico xml se presentan en la tabla 1. En ella se puede ver que el tópico *SVG* está relacionado con los tipos de tópicos: "lenguaje de etiquetas", "vocabulario xml" y "formato gráfico"[13] mientras que el tópico xml tiene como tipo de tópicos general: "lenguaje de etiquetas". Esto se halla codificado en las siguientes líneas de código:

Tabla I

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<topicmap id="articulo_svg"
                    xmlns="http://www.topicmaps.org/xtm/1.0/"
                    xmlns:xlink="http://www.w3.org/1999/xlink"
>
<topic id="svg">
          <instanceof>
<topicref xlink:href="#lenguaje_de_etiquetas"/>
</instanceof>
          <instanceof>
<topicref xlink:href="#vocabulario_xml"/>
</instanceof>
          <instanceof>
<topicref xlink:href="#formato_grafico"/>
</instanceof>
     <!--version: 1.1 -->
     <!--status: recomendaci n del w3c -->
     <basename>
          <basenamestring>scalable vector graphics</basenamestring>
          <variant>
               <parameters><topicref xlink:href="#acronimo"/></parameters>
               <variantname><resourcedata>svg</resourcedata></variantname>
          </variant>
     </basename>
     <occurrence>
          <instanceof><topicref xlink:href="#articulo"/></instanceof>
          <scope><topicref xlink:href="#offline"/></scope>
          <resourceref
xlink:href=hyperlink"file:///c|/articulos/svg_2.doc"/>
     </occurrence>
     <occurrence>
          <instanceof><topicref xlink:href="#articulo"/></instanceof>
          <scope><topicref xlink:href="#online"/></scope>
          <resourceref
xlink:href=hyperlink"http://www.blabla.bla/svg2.html"/>
     </occurrence>
  </topic>
</topicmap>

el t pico xml podr a recogerse en el siguiente c digo:

<topic id="xml">
     <instanceof>
<topicref xlink:href="#lenguaje_de_etiquetas"/>
</instanceof>
   <!--version: 1.0 -->
   <!--status: recomendaci n del w3c -->
   <basename>
     <basenamestring>extensible markup language</basenamestring>
     <variant>
               <parameters><topicref xlink:href="#acronimo"/></parameters>
               <variantname><resourcedata>xml</resourcedata></variantname>
     </variant>
   </basename>
   <occurrence>
     <instanceof><topicref xlink:href="#articulo"/></instanceof>
     <scope><topicref xlink:href="#offline"/></scope>
     <resourceref xlink:href=hyperlink"file:///c|/articulos/svg_2.doc"/>
   </occurrence>
   <occurrence>
     <instanceof><topicref xlink:href="#articulo"/></instanceof>
     <scope><topicref xlink:href="#online"/></scope>
     <resourceref xlink:href=hyperlink"http://www.blabla.bla/svg2.html"/>
</occurrence>
   <occurrence>
     <instanceof><topicref xlink:href="#recomendacion"/></instanceof>
     <scope><topicref xlink:href="#online"/></scope>
     <resourceref xlink:href=hyperlink"http://www.w3.org/xml/"/>
</occurrence>
</topic>
```

```
<topic id="xml">
<instanceof>
<topicref xlink:href="#lenguaje_de_etiquetas"/>
</instanceof>
</topic>
```

Además, dentro de ambos tópicos se da una serie de elementos <occurrence> que sirven para relacionar el tópico en cuestión con los recursos de información que le son relevantes.

Una de las posibles asociaciones entre estos dos tópicos constituiría a su vez un nuevo tópico, <association>, perteneciente al tipo de tópicos "#vocabulario_de" ya que *SVG* es el vocabulario de la sintaxis xml para gráficos bidimensionales. En el código se puede apreciar que el papel de *SVG* en esta asociación es el de "#vocabulario" y el de xml es el de "#sintaxis".

```
< association >
< instanceof ><topicref xlink:href="#vocabula-
rio_de"/></ instanceof >
     < member >
     < rolespec ><topicref xlink:href="#vocabula-
rio"/></ rolespec >
     < topicref xlink:href="#svg"/>
     </ member >
     < member >
     < rolespec ><topicref xlink:href="#sinta-
xis"/></ rolespec >
     < topicref xlink:href="#xml"/>
     </ member >
</ association >
```

De este modo se podrían seguir añadiendo tópicos, tipos de tópicos, asociaciones y ocurrencias hasta alcanzar el nivel descriptivo deseado. En otras palabras, hasta alcanzar la indización adecuada con vistas a la posterior recuperación. El modelo de RI asociado a este tipo de descripción es, necesariamente, más sofisticado que los que existen en la actualidad. Este modelo teórico debe tener en cuenta, entre otras cosas, que los metadatos no son aplicables exclusivamente a recursos textuales sino que se pueden utilizar con objetos digitales de cualquier tipo. En el caso de *SVG*, su dualidad texto-imagen y el hecho de que su parte textual siga la sintaxis xml, le convierten en un candidato ideal para desarrollar experimentos en este sentido.

Figura 2

Problemas asociados al uso de metadatos. El principal problema del uso de metadatos como método para describir el contenido y características de la información gráfica es el mismo que se da cuando se trata de contexto textual: ¿quién asigna los metadatos?, ¿qué programas son capaces de interpretarlos o realizar búsquedas basándose en ellos? La tarea de automatizar el proceso de asignación y gestión de metadatos sigue siendo una utopía. Uno de los primeros desafíos es encontrar una forma de integrar diferentes representaciones o, cuando menos, acceder eficazmente a ellas desde un punto de partida común. *SVG* puede ofrecer nuevas posibilidades para la información gráfica en este sentido.

2.3. Posibles escenarios para la recuperación de contenido gráfico en la web

La infraestructura de comunicaciones que hoy en día está al alcance de los usuarios permite que puedan acceder rápidamente a la información gráfica. Esto ha originado la necesidad de hacer más eficiente la recuperación de ese tipo de materiales. Como ya se ha comentado, los interfaces de búsqueda en la web no proporcionan todavía un acceso eficaz a este tipo de información.

Sin embargo, existen algunos casos en los que ese acceso constituye una parte importante de cualquier inversión en automatización, como por ejemplo en el campo de la medicina.

El personal médico utiliza gráficos constantemente. Un ejemplo concreto son los electrocardiogramas (figura 2). Estos pueden ser presentados en forma de texto o cifras, sin embargo, los médicos recurren casi unánimemente al formato gráfico a la hora de consultarlos.

Supongamos que en un hospital se propone un sistema de almacenamiento y recuperación de este tipo de objetos gráficos. La utilidad de un sistema así está fuera de toda duda. Pero ¿qué tipo de interfaz RI sería el adecuado?, ¿qué tipo de indización?

Al consultar un electrocardiograma, un médico asocia un tipo de patrón de la representación gráfica con la correspondiente situación clínica. La indización de los objetos digitales tendría que tener presente esta relación. El usuario necesita recuperar información usando dos tipos de parámetros: patrones gráficos (recuperación visual) y descriptores clínicos (recuperación textual).

El índice, por lo tanto, debería ser capaz de registrar y combinar estos dos tipos de informaciones: las características gráficas del objeto en sí, como por ejemplo la longitud de las ondas cardíacas, su frecuen-

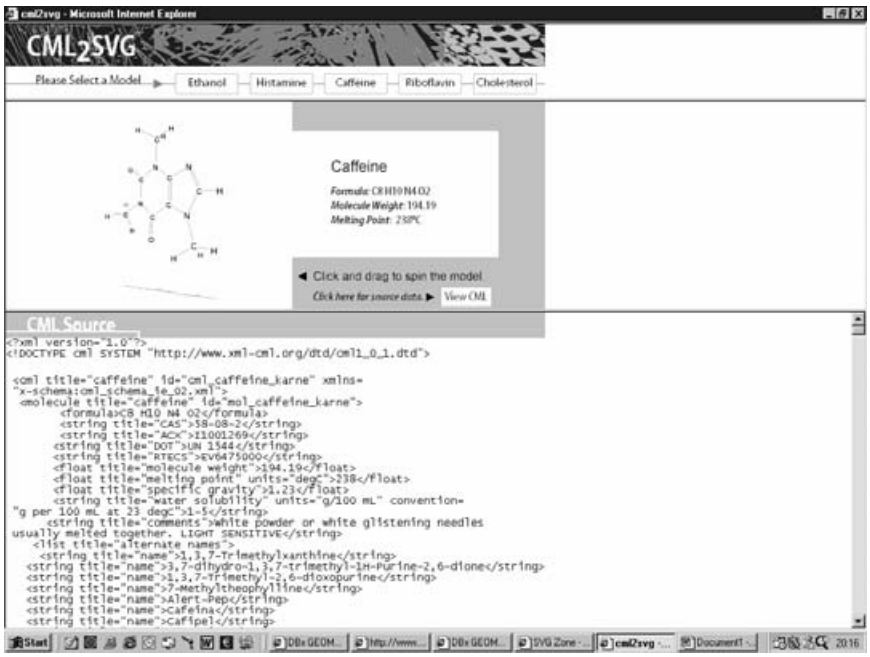


Figura 3

cia, la amplitud de cada sección, etc., y la información clínica correspondiente.

En cuanto a la interfaz, sería un híbrido que posibilitaría simultáneamente las consultas visual y textual. La consulta visual se formalizaría en un gráfico con las características informativas que se desee recuperar: por ejemplo, secciones de un electrocardiograma con ondas de una determinada longitud y frecuencia. Por descontado, la elaboración de esa consulta gráfica sería asistida por la propia interfaz. La consulta textual se basaría en descriptores clínicos.

La dualidad texto-imagen de *SVG* lo convierte en un formato idóneo para este tipo de casos. A la hora de elaborar un índice, se podrían asociar metadatos al objeto digital con el fin de describirlo para su posterior recuperación, combinando de esta manera lo gráfico (*SVG*) con lo clínico (metadatos). De hecho, la propia especificación *SVG* cuenta con el elemento "metadata" para introducir descripción sobre el contenido del objeto. Esta información se puede cumplimentar utilizando determinadas propiedades propias de la especificación, o bien utilizando otros sistemas de metadatos (*DC*, rdf...) siempre y cuando hayan sido declarados convenientemente como namespace al principio del documento.

http://www.w3.org/TR/SVG11/metadata.html

En cuanto a la interfaz gráfica, es posible plantearse la elaboración y proceso de una consulta gráfica en formato *SVG*, pero sería impensable hacer lo mismo con una imagen gif. Por otra parte, los algoritmos utilizados para comparar una consulta textual con un índice textual se podrían adaptar con relativa facilidad al escenario de un sistema de búsqueda *SVG*, mientras que la comparación de imágenes "puras" exigiría un entorno completamente distinto.

La medicina no es el único campo donde un sistema de RI gráfica comienza a ser una necesidad; otras áreas que se pueden citar son: arquitectura, arqueología, química, biología, astronomía, ingeniería, cartografía y cualquier tipo de sistema automatizado que incluya mapas, etc.

La figura 3 es una demostración[14] de la combinación de *SVG* con el lenguaje de marcado químico *CML*[15]. En la parte superior izquierda de la imagen se puede ver la representación gráfica de una molécula de cafeína. En la parte superior derecha la descripción de dicha molécula. En la parte inferior el código *CML* correspondiente a esa molécula desde el que se ha generado la representación gráfica mediante una transformación[16]. Hay varias cuestiones importantes que se pueden destacar en este ejemplo:

—La combinación de dos vocabularios de la sintaxis xml: *SVG* (gráfico) y *CML* (químico) en un solo sistema de gestión de información.

—El hecho de que se pueda transformar un objeto textual (el código *CML* de la parte inferior de la imagen) en un objeto gráfico (el gráfico de la parte superior izquierda). Ese proceso de transformación se puede automatizar usando el lenguaje *Xslt* (*extensible stylesheet language transformations*).

—Aunque *SVG* es especialmente adecuado para gráficos bidimensionales, se puede utilizar también para simular gráficos tridimensionales como en el ejemplo de la figura 3[17].

—Los gráficos *SVG* pueden animarse; en el ejemplo se da la posibilidad de rotar la imagen.

La conclusión que se puede sacar de una demostración como la que se cita es que la implementación de sistemas de RI gráfica a gran escala sólo es una cuestión de tiempo.

3. *SVG*: texto e imagen al mismo tiempo

Como se ha venido repitiendo a lo largo del artículo, *SVG* es un vocabulario xml por medio del cual se formaliza un conjunto de elementos gráficos como rectángulos, líneas, polígonos, elipses, etcétera. De la combinación de esos elementos surgen los documentos *SVG*. La característica más destacada de esos documentos es que son a la vez texto e imágenes. Ponga-

mos un ejemplo, el siguiente código *SVG* se corresponde con el gráfico de la figura 4.

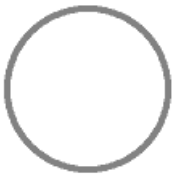


Figura 4

```
<?xml version="1.0" standalone="no"?>
<svg      width="400px"
height="250px"
preserveAspectRatio="xMidYMid meet"
xmlns="http://www.w3.org/2000/svg"
zoomAndPan="magnify"
>
<circle cx="60px"
cy="100px"
r="50px"
style="fill:white; stroke:red; stroke-width:4"
/>
</svg>
```

En el código se define un elemento círculo ("circle"), las coordenadas "x" e "y" de su centro (atributos "cx" y "cy") en píxeles, la longitud también en píxeles de su radio (atributo "r") y la forma en que va a presentarse (atributo "style"). En otras palabras: se está escribiendo un gráfico. Los navegadores dotados de un plugin *SVG* reconocerán el código que aparece a la izquierda y presentarán la imagen que aparece a la derecha.

Esta ambivalencia texto-imagen es lo que convierte a *SVG* en un formato ideal para servir de puente entre la información textual y la gráfica.

3.1. Caracteristias del formato *SVG*

En un apartado anterior se mencionaba *Flash* de *Macromedia* como el formato de gráficos vectoriales más utilizado y conocido en la web. El inconveniente

Figura 6. Los gráficos SVG, como en este ejemplo de un mapa del oeste de los EUA pueden generarse dinámicamente en un servidor a partir de los datos solicitados por un visitante. http://www.dbxgeomatics.com/svgmapmaker/samples/west_usa.htm

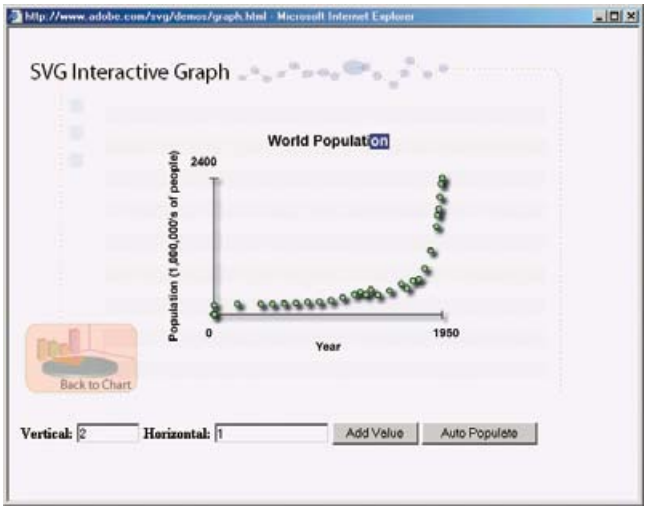


Figura 5. El texto contenido en un gráfico SVG es editable: puede ser seleccionado, buscado, indizado, etc.

de este producto es que pertenece a una compañía: *Macromedia* que puede modificarlo o no según sus intereses. Por otra parte, se trata de un formato totalmente extrínseco con respecto a html o xml. Por estas razones el *World Wide Web Consortium* tomó la iniciativa de crear un formato vectorial abierto y estándar: el *SVG*[18].

Algunas de las características de este formato son:

—Tiene todas las cualidades asociadas a un formato vectorial: es escalable, compacto, con formas editables mediante curvas *Bézier*, con contornos suavizados, transparencias y capaz de incluir mapas de bits.

—El tamaño de los ficheros *SVG* es muy compacto.

—El texto es editable, admitiendo las fuentes escalables más comunes, como *TrueType* o *Type 1*. Esto supone una ventaja sobre los formatos gif o jpg: el texto que contiene un gráfico *SVG* puede ser editado, seleccionado, usado como referencia para un enlace, indexado por los buscadores, etc. (figura 5).

—La calidad de colores es excelente. El color del gráfico se puede calibrar con los sistemas estándar de gestión de color.

—El fichero *SVG* no es binario: se trata de un fichero de texto normal y corriente. Esto significa que se puede editar con cualquier procesador de textos y sus contenidos se pueden indexar, buscar, etc.

—Es compatible con los estándares actuales de la web.

En primera línea en Sistemas de Información y Gestión del Conocimiento.

- Informática Documental
- Internet, Intranet
- Edición de Bases de Datos en CD-Rom
- Sistema Integrado de Gestión Bibliotecaria Absys
- Catalogación Retrospectiva
- Sistema Integrado de Gestión de Centros Archivísticos Albalá

baratz

Raimundo Fernández Villaverde, 28
28003 Madrid (España)
Teléfono +34 91 456 03 60 - Fax +34 91 533 09 58
www.baratz.es - E-mail: informa@baratz.es

—Incluye soporte para hojas de estilo *CSS* (*cascading style sheets*). Esto significa que con las hojas de estilo será posible modificar en cascada no sólo texto, sino también gráficos en una colección de ficheros.

—Puede incluir código para modificar el gráfico dinámicamente.

—Es xml, y xml es un formato extensible. Los fabricantes de software empiezan a adoptarlo como formato nativo para sus aplicaciones. La consecuencia es que *SVG* va a ser comprensible por cualquier aplicación que reconozca el formato xml.

—Admite efectos de sonido, visuales, eventos asociados al ratón, etiquetas informativas, etc.

—Puede generarse dinámicamente en un servidor web como respuesta a instrucciones de *Java*, *JSP*, *JavaScript*, *Perl*, *Plsql*, *ASP*, *Xslt*, etcétera. Por ejemplo, pueden crearse en tiempo real gráficos con las cotizaciones de bolsa o mapas interactivos como el de la figura 6.

—Un fichero *SVG* tiene un tamaño menor que sus equivalentes codificados en mapa de bits como ficheros gif o jpg. Por ejemplo, el reloj *SVG* de la figura 7 es perfectamente funcional gracias a la combinación de *JavaScript* y *SVG* y ocupa 2K.

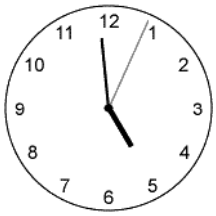


Figura 7

—Los gráficos *SVG* son independientes de la resolución, de modo que su tamaño puede ser aumentado o reducido mediante el zoom sin que la calidad de la imagen resulte deteriorada.

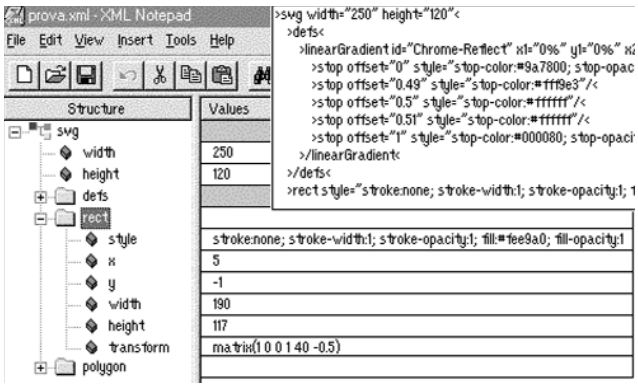


Figura 8. A diferencia de otros formatos gráficos, el contenido de un archivo SVG es texto normal. Esto significa que puede editarse incluso con un bloc de notas. Aquí tenemos un ejemplo de código SVG editado con el XML Notepad de Microsoft. Los diferentes elementos del gráfico se identifican en cada carpeta (por ejemplo rect, polygon, etc.) y los parámetros figuran al lado

—Los objetos gráficos, al estar expresados en xml, pueden ser etiquetados textualmente. Esta circunstancia constituye una gran ventaja a la hora de su posterior indización y recuperación (figura 8). De hecho, los documentos *SVG* pueden ser indizados por los motores de búsqueda de la web y, por lo tanto, la información que contienen es directamente recuperable como en cualquier otro documento xml.

—Por la misma razón, los objetos gráficos o sus componentes pueden ser enlaces a partes de la misma imagen, otras imágenes, páginas web, etc.

—Con *SVG* se pueden realizar animaciones con cualquier grado de complejidad.

El siguiente ejemplo (figura 9) ilustra la escalabilidad de *SVG* (formato vectorial) en comparación con *PNG* (mapa de bits). Cuando se amplía un mapa de bits la imagen resultante experimenta una pérdida de calidad significativa. En cambio, si se trata de una imagen vectorial la calidad no cambia.

Figura 9. La imagen central muestra la ampliación del primer fichero en PNG, mientras que la de la derecha es la ampliación del mismo fichero en formato SVG, la diferencia de calidad es obvia

En cuanto al alcance de la implementación de *SVG* en la web, excepto algunos "builds" de *Mozilla*, ningún navegador es capaz todavía de reconocer y presentar directamente gráficos en este formato. Como ocurre con *Flash*, los navegadores no han desarrollado parsers[19] propios para el código SVG, de forma que es necesario instalar un plugin. En la siguiente lista se dan los nombres de algunas de las compañías más activas en cuanto a la implementación de SVG junto al software que producen:

—*Adobe SVG support*. Además de desarrollar uno de los plugins más utilizados para visualizar *SVG*, el *Adobe SVG viewer* (actualmente en su versión 3.0), produce otros programas capaces de editar *SVG* como *Illustrator 10* y *LiveMotion*:
http://www.adobe.com/svg/

—*Corel* ofrece un plugin para navegadores junto a una serie de filtros para exportar información *SVG* dentro de su programa *CorelDraw 9*. A partir de la versión 10 estos filtros están integrados dentro del software:

http://venus.corel.com/nasapps/DrawSVGDownload/index.html

—El *W3C*, por su parte, ofrece un programa open-source capaz de procesar y analizar sintácticamente documentos *SVG*:
http://koala.ilog.fr/jackaroo/

—*ePicture* es el nombre de la empresa que creó *Beatware*, programa para *Mac* que permite exportar ficheros vectoriales a formato *SVG*:
http://www.beatware.com/

—*IBM* también dispone de un plugin *SVG* para navegadores:
http://www.alphaworks.ibm.com/tech/svgview

—*ImageMagik* es un conjunto de librerías para gestionar información gráfica en diferentes formatos, entre ellos *SVG*:
http://www.imagemagick.org/

—*Jasc* ha desarrollado un programa, *WebDraw* —antes conocido como *Trajectory Pro*— que permite generar gráficos *SVG*:
http://www.jasc.com/products/webdraw/

—X-Smiles cuenta con un visor gratuito. En la actualidad va por la versión 0.71:
http://www.x-smiles.org/download.html

Se puede afirmar, en conclusión, que *SVG* es un formato con excelentes perspectivas de crecimiento en la web.

3.2. Características del vocabulario *SVG*

La idea más importante es que los elementos gráficos en *SVG* siguen siendo elementos xml. El elemento <svg>, por ejemplo, es el elemento raíz de un gráfico *SVG*. Dicho de otro modo, es el "contenedor" de mayor jerarquía que incluye a todos los demás elementos de ese gráfico. Dentro de ese elemento, se pueden incluir en general tres tipos de entidades visuales:

—Formas gráficas

—Texto

—Mapas de bits

Formas gráficas: Los elementos de forma básicos en SVG son:

—Rectángulos: <rect/>

—Círculos: <circle/>

—Elipses: <ellipse/>

—Líneas: <line/>

—Polilíneas: <polyline/>

—Polígonos: <polygon/>

—Ruta: <path/>

Se ha mencionado el elemento círculo al comienzo del apartado tres, a continuación se comentarán brevemente algunos[20] de los demás elementos. El siguiente código dibuja un rectángulo (figura 10):

Figura 10

```
<?xml version="1.0" standalone="no"?>
<svg width="300" height="300">
<rect     x="80" y="53" width="189" height="52"
                    style="
fill:rgb(39,44,231);
                          stroke:rgb(0,0,128);
                          stroke-width:1"/>
</svg>
```

La tercera línea del código define el rectángulo. La definición comienza declarando el elemento con la etiqueta <rect>. Los atributos "x" e "y" son coordenadas que se utilizan para dar una posición absoluta al elemento en el navegador: si se sustituye por ejemplo 80 por 0 el rectángulo se situará a 0 píxeles del margen izquierdo del navegador. La declaración de estilo permite definir propiedades *CSS*[21] que se aplicarán al elemento. La propiedad "fill", por ejemplo, se refiere al color con el que se va a rellenar, en este caso azul.

A continuación se define un polígono (figura 11):

Figura 11

```
<?xml version="1.0" standalone="no"?>
<svg width="100%" height="100%">
<polygon  points="223.5,123.034 276,213.966
171,213.966"
                    style="
fill:rgb(126,14,83);

stroke:rgb(0,0,128);
                              stroke-width:1"/>
</svg>
```

Los polígonos se usan en SVG para definir formas que contengan al menos tres lados. En ejemplo, la forma del polígono se define mediante una serie de puntos dobles. Por ejemplo, un lado del triángulo se puede representar con los valores 179,33. El 179 se refiere a la posición "x" de esa línea del triángulo y 33 se refiere a la posición "y" de la misma línea. El control sobre la presentación de los elementos es, como se ve, absoluto.

Por ultimo, veamos un ejemplo del elemento ruta (*path*) (figura 12):

```
<?xml version="1.0" standalone="no"?>
<svg width="100%" height="100%">
   <path  d="M10 20 L110 20 L110 120 L10 120"
```

```
                style="   fill:rgb(0,0,153);
fill-opacity:0.5;
stroke:rgb(0,0,153);
stroke-width:4"/>
</svg>
```

Figura 12

El elemento *path* es probablemente la forma más compleja en *SVG*, pero también la más versátil y utilizada. El atributo "d" permite la descripción de distintos tipos de dibujos. Debemos imaginarnos que estamos trazando una línea con un lápiz. M10 20, lo que significa mover la punta del lápiz a las coordenadas 110,20 en el "lienzo" *SVG* del navegador. *Path* usa una forma más críptica de codificación que el resto de elementos *SVG*. La razón es que así es posible definir formas muy complejas dentro de un solo elemento. Por ejemplo, el siguiente código:

```
<svg width = "500" height = "500" viewBox = "0 0
110 110">
<g stroke = "black" fill = "white" stroke-width =
"0.3">
<path d = "M 2 2 h 100 v 99 h -98 v -97 h 96 v
95 h -94 v -93 h 92 v 91 h -90 v -89 h 88 v 87 h
-86 v -85 h 84 v 83 h -82 v -81 h 80 v 79 h -78
v -77 h 76 v 75 h -74 v -73 h 72 v 71 h -70 v -
69 h 68 v 67 h -66 v -65 h 64 v 63 h -62 v -61 h
60 v 59 h -58 v -57 h 56 v 55 h -54 v -53 h 52 v
51 h -50 v -49 h 48 v 47 h -46 v -45 h 44 v 43 h
-42 v -41 h 40 v 39 h -38 v -37 h 36 v 35 h -34
v -33 h 32 v 31 h -30 v -29 h 28 v 27 h -26 v -
25 h 24 v 23 h -22 v -21 h 20 v 19 h -18 v -17 h
16 v 15 h -14 v -13 h 12 v 11 h -10 v -9 h 8 v 7
h -6 v -5 h 4 v 3 h -2 v -1 "></path>
</g>
</svg>
```

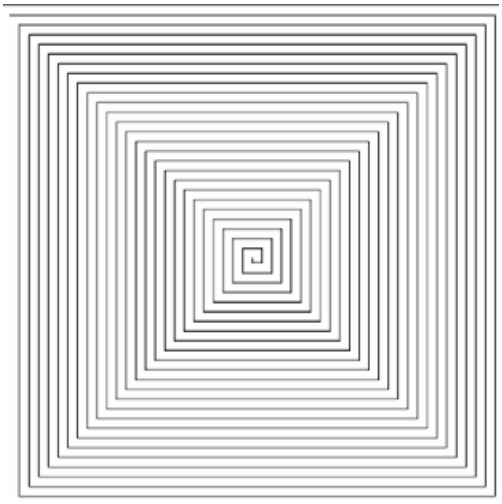


Figura 13

Da como resultado el gráfico que se muestra en la figura 13. Este gráfico es relativamente simple en términos de lo que se puede conseguir con *SVG*. Algunos de los posibles valores del atributo "d" son[22]: M = mover hacia, L = línea hacia, H = línea hacia horizontal, V = línea hacia vertical, C = curva hacia, Q = curva bezier, A = arco elíptico, Z = cierre de *path*.

Debido a la complejidad de los *paths*, es recomendable utilizar programas especializados para describirlos como por ejemplo el software *WebDraw* de *Jasc*.

Texto: Lo más importante a destacar sobre el texto como entidad visual dentro de un gráfico *SVG* es que sigue siendo texto y, por lo tanto, puede ser indexado y recuperado por cualquier motor de búsqueda textual. También, como las formas gráficas, las imágenes y el resto de elementos *SVG*, es susceptible de ser enlazado con otros recursos mediante el uso del elemento <a> en combinación con la sintaxis *XLink*[23]. Un enlace en *SVG* se podría expresar de la siguiente forma:

```
<a xlink:ref.=http://www.xmml.com>
     <cualquier tipo de elemento>
</a>
```

Mapas de bits: *SVG* es especialmente apropiado para representar gráficos bidimensionales como mapas o diagramas, pero se puede combinar con otros tipos de media. Es posible, por ejemplo, anidar mapas de bits en un documento *SVG*. Para ello se utiliza el elemento <image>. Los browsers *SVG* soportan, como mínimo los formatos png y jpg.

```
<image x=   y=   width=   height=
xlink:href= imagen.png >
```

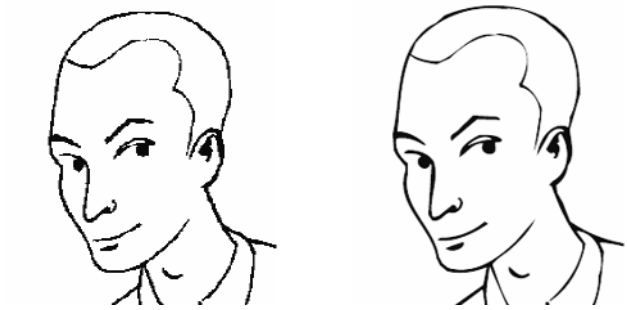


Figura 14. A la izquierda, el fichero original en formato PNG. A la derecha, el mismo fichero convertido a SVG con el programa CR2V

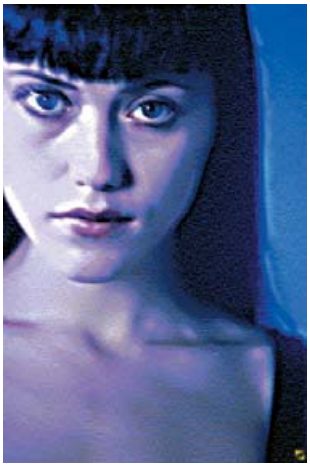
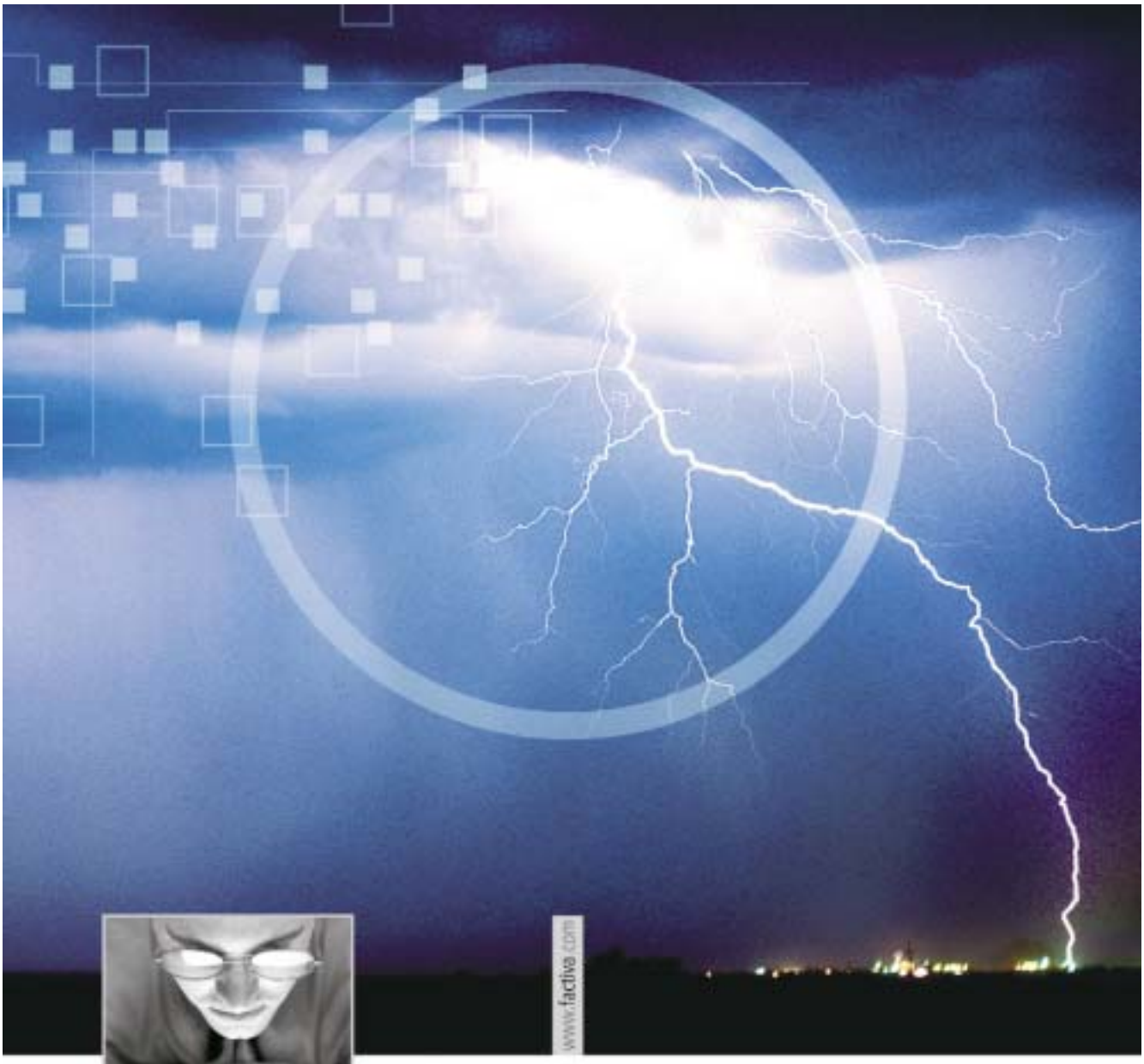


Figura 15

Figura 16



La inspiración solamente llega una vez. No dos.

Para toda decisión importante no se tiene normalmente una segunda oportunidad. Cuando llega el momento de la verdad, usted necesita acceso inmediato a una información precisa y fiable.

Para responder eficazmente, Factiva le da acceso a información profesional y económica, local e internacional, para inspirarle en sus mejores decisiones. Cuando usted lo necesita, como usted lo necesita.

Para más información, visite **www.factiva.com/es/inspiration**

factiva
Dow Jones & Reuters

inspiring business decisions

Pero además, también es posible convertir mapas de bits en gráficos vectoriales. Uno de los programas que se dedica a ello es *CR2V*. Los resultados son variables con gráficos y fotografías (figura 14). A la izquierda el original de un gráfico *PNG*, a la derecha el resultado de la conversión en *SVG*. Pero la mayor diferencia la apreciamos en el tamaño. El fichero de la figura 15 —en *PNG*— ocupa 180 KB, mientras que el mismo, en *SVG*, algo menos de 16 KB (figura 16).

En el caso de la fotografía, el ahorro en espacio es evidente y la pérdida de calidad también. Lo realmente interesante es que el código de la imagen de la derecha se compone exactamente de 240 elementos: el elemento introductorio <xml>, el elementos raíz <svg> y 238 elementos <path>. Esto significa que las fotografías también pueden convertirse en conjuntos de elementos textuales susceptibles, como se ha visto, de ser indizados y por lo tanto, de ser integrados en sistemas RI.

A lo largo de todo el artículo se ha insistido en que SVG es en sí mismo un lenguaje descriptivo debido a dos motivos. Por una parte obedece las reglas de la sintaxis xml y por lo tanto es capaz de explicitar la estructura del contenido de un documento digital. Por otra parte constituye un vocabulario creado para describir explícitamente una serie de elementos gráficos y sus propiedades.

Pongamos un ejemplo: una imagen *SVG* se compone de una serie de elementos agrupados en una jerarquía[24]. Cada uno de ellos puede tener un título y una descripción. Atendiendo a la jerarquía es posible crear un esbozo mental de la imagen que el documento describe sin haberla visto todavía. Esto implica que la imagen posee una estructura que, en teoría, se podría usar para fines de indización y recuperación del objeto que describe.

Esta estructura se formaliza en el documento mediante el elemento <g>, con el cual es posible agrupar otros elementos siguiendo diferentes criterios. Pongamos un ejemplo.

El siguiente documento *SVG* representa la imagen de un barrio. Dentro del barrio se describe una serie de elementos: dos calles, un complejo deportivo y un parque. Cada uno de esos cuatro subcomponentes puede, a su vez, contener otros subcomponentes, entre ellos un elemento título y un elemento descripción. En el ejemplo no se describen todavía los componentes gráficos de la imagen y, sin embargo, el observador ya se puede hacer una idea mental del objeto:

```
<?xml version="1.0"?>
<svg width="6cm" height="4.5cm" viewBox="0 0 600
450">
<title>Descripci n gr fica del barrio Ooster-
park</title>
<desc>Ejemplo de distribuci n urban stica en Gro-
ningen</desc>
<g id="barrio_Oosterpark">
<title>Barrio Oosterpark</title>
<desc>Barrio popular holandes de la decada 1930-
1940</desc>
<g id="calle_Hortensialaan">
<title>Hortensialaan</title>
<desc>La basura se saca los mi rcoles</desc>
</g>
<g id="calle_Begoniastraat">
<title>Begoniastraat</title>
<desc>Calle peatonal</desc>
</g>
<g id="parque_Pioenpark">
<title>Pioenpark</title>
<desc>Dos lagunas y un campo de f tbol</desc>
</g>
<g id="complejoDeportivo_Denksport">
<title>Denksport</title>
<desc>Complejo deportivo y cultural</desc>
<g id="gimnasio_HenkMeijer">
<title>TaeKwonDo School Henk Meijer</title>
<desc>Abierto lunes, martes y viernes</desc>
</g>
</g>
     </g>
</svg>
```

Definir el contenido de un documento o, como en el ejemplo, de un gráfico mediante grupos de elementos es útil por distintos motivos. Lo más interesante de cara a la RI es el hecho de que los distintos grupos pueden ser gestionados por separado para producir, por ejemplo, distintos índices. En el código de la fotografía que se ha mostrado antes sería posible agrupar los elementos <path> siguiendo distintos criterios: elementos relativos al fondo o a la figura, curvos o rectilíneos, según el color, etc.

El siguiente paso sería asignar metadatos a documento, elementos individuales y grupos de elementos para conseguir una imagen "autodescriptiva". Si se observa el código anterior, tenemos ya la descripción de una estructura (xml) y un tipo básico de metadatos (los elementos <title> y <desc>). Lo que *SVG* aporta es la posibilidad de importar en el documento información relativa a otros ámbitos utilizando las referencias conocidas como namespaces[25]. Pongamos un ejemplo, digamos que hemos creado un hipotético vocabulario xml sobre arquitectura y queremos asociarlo a nuestro gráfico SVG:

```
<g>
<title>Ventana del dormitorio</title>
     <g>
          <desc
xmlns:arquitectura="http://www.bla.com/arquitec-
tura/ventanas">
               <arquitectura:material>aluminio</ar-
quitectura:material>
               <arquitectura:dimensiones altu-
ra="113" anchura="170"/>
          </desc>
          <use xlink:href="ventana"
transform="translate(110, 160)"/>
     </g>
</g>
```

De esta misma forma se podría hacer uso de auténticos lenguajes de metadatos, como rdf, *DC* o *XTM*. Un ejemplo con *DC* y rdf:

```
<g>
<title>Ventana del dormitorio</title>
     <g>
          <desc
xmlns:arquitectura="http://www.bla.com/arquitec-
tura/ventanas">
               <arquitectura:material>aluminio</ar-
quitectura:material>
               <arquitectura:dimensiones altu-
ra="113" anchura="170"/>
          </desc>
          <rdf:RDF
xmlns:rdf="http://www.w3c.org/1999/02/22-rdf-syn-
tax-ns#"

xmlns:dc="http://purl.org/dc/elements/1.1">
<rdf:Description
rdf:about="http://ejemplo.org/ventanas.svg">
     <dc:creator>P rez L pez</dc:creator>
     </rdf:Description>
</rdf:RDF>
<use xlink:href="ventana" transform="transla-
te(110, 160)"/>
</g>
</g>
```

Por supuesto, para hacer uso de toda esta información en un entorno RI, son necesarios agentes capaces de extraerla e interpretarla, pero las posibilidades son innegables.

3.3. Transformaciones *Xslt* para generar o transformar *SVG*

En el apartado 2.3 se describía un sistema en el que se integraban dos tipos de vocabularios xml: *SVG* y *CML*. Los gráficos *SVG* que se presentaban en ese sistema eran el resultado de la transformación del código *CML*. A su vez, en el apartado 2.2 se hablaba de otro vocabulario xml especializado en metadatos: los mapas de tópicos o *XTM*. El código de un mapa de tópicos es tan susceptible de ser transformado en un gráfico como el de un documento *CML*. Las posibilidades que plantea esa idea sobre la combinación de *SVG* y metadatos son muy interesantes.

Para realizar este tipo de combinaciones y transformaciones la herramienta ideal es el lenguaje *Xslt*[26]. Su funcionamiento es sencillo: un parser *Xslt* analiza el documento fuente[27] y, siguiendo las instrucciones en la hoja de estilo *Xslt*, transforma los fragmentos xml especificados.

Como ejemplo, una transformación sencilla comparable a la que se refiere en el apartado 2.3. En este caso se va a transformar[28] el documento textual xml resumenes.xml que se presenta a continuación:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<resultados intervalo="10" valormaximo="100">
<item nombre="enero">30</item>
     <item nombre="febrero">50</item>
     <item nombre="marzo">70</item>
     <item nombre="abril">10</item>
     <item nombre="mayo">100</item>
     <item nombre="junio">80</item>
     <item nombre="julio">25</item>
     <item nombre="agosto">60</item>
     <item nombre="septiembre">10</item>
     <item nombre="octubre">100</item>
     <item nombre="noviembre">90</item>
     <item nombre="diciembre">30</item>
</resultados>
```

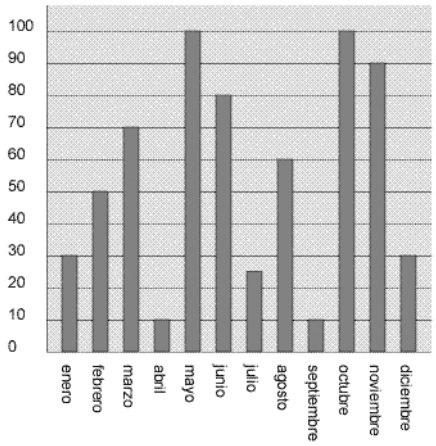


Figura 17

Gráfico *SVG* resumenes.svg que se puede ver en la figura 17. Para generar el mismo sólo es necesario utilizar tres de los elementos SVG más básicos:

<RECT></RECT>, para definir rectángulos.

<LINE></LINE>, para trazar líneas.

<TEXT></TEXT>, para el texto.

La hoja de estilo *Xslt* empieza con la demarcación de una serie de variables:

```
<xsl:variable name="yMargenTop">50</xsl:variable>
<xsl:variable name="yMargenBottom">100</xsl:va-
riable>
<xsl:variable name="xMarginLeft">50</xsl:varia-
ble>
<xsl:variable name="xMarginRight">50</xsl:varia-
ble>
<xsl:variable name="yGrafica">
     <xsl:value-of select="$yVentana-$yMargenTop-
$yMargenBottom"/>
</xsl:variable>
```

En estas variables se define y calcula:

—El tamaño de la ventana que va a contener el gráfico.

—Los márgenes horizontales y verticales.

—El número de columnas que se van a trazar y el espacio que ocuparán, etc.

A continuación se construye el fichero *SVG* en la regla de construcción de resultados.

```
<xsl:template match="resultados">
  <svg width="{$xVentana}" height="{$yVentana}">

<!--Trazamos el rect ngulo que contendr  el gr -
fico-->
```

```
    <rect x="{$xMarginLeft}"
          y="{$yMargenTop - 20}"
          width="{$xGrafica}"
          height="{$yGrafica + 20}"
          style="fill:#E3DFCC;"/>

<!--Trazamos las l neas verticales-->

    <xsl:call-template name="plvertical"/>

<!--Trazamos las l neas horizontales y el texto a
su lado-->

    <xsl:call-template name="plhorizontales"/>

<!--Procesamos el elemento item para pintar las
columnas-->

    <xsl:apply-templates select="item"/>

  </svg>
</xsl:template>
```

El primer paso es dibujar un rectángulo de color gris que será el fondo sobre el que se va a trazar el gráfico.

```
<!--Trazamos el rect ngulo que contendr  el gr -
fico-->

    <rect x="{$xMarginLeft}"
          y="{$yMargenTop - 20}"
          width="{$xGrafica}"
          height="{$yGrafica + 20}"
          style="fill:#E3DFCC;"/>
```

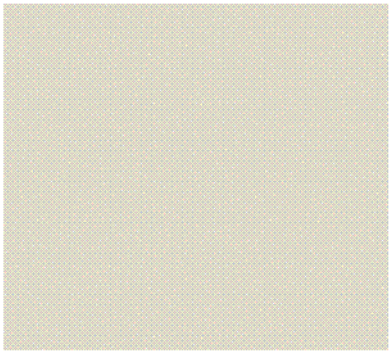


Figura 18

Ese código da como resultado el gráfico de la figura 18. Después se define la línea del eje Y (vertical) mediante la regla de construcción *plvertical*, invocada por medio de la instrucción XSLT: <xsl:call-template name="plvertical"/> :

```
<xsl:template name="plvertical">
 <line x1="{$xMarginLeft}"
       y1="{$yMargenTop - 20}"
       x2="{$xMarginLeft}"
       y2="{$posyGrafica}"
       style="stroke:#000000; stroke-width:0.1"
/>
</xsl:template>
```

A continuación, y mediante las reglas de construcción *plhorizontales*, *plhorizontales1*, *plhorizontales2* y *ptextoy*, invocadas por medio de la instrucción *Xslt* <xsl:call-template name="plhorizontales"/>, se define el eje horizontal (x) y el resto de las líneas horizontales con la numeración que corresponde. La figura 19 muestra el resultado.

Figura 19

Después se deben procesar los elementos <item> del documento resumenes.xml para trazar las barras/rectángulos que corresponderán a cada mes en el gráfico final junto al texto que les acompaña. Esto se hace mediante las reglas de construcción *pintarRectangulo* y *ptextox*. Con esto se obtiene el resultado final (figura 17). En la tabla 2 se puede observar el código SVG de este gráfico, generado mediante la combinación de resumenes.xml, así como la hoja de estilo que se acaba de analizar (xmltosvg.xsl).

Lo que se ha pretendido demostrar en este apartado mediante un ejemplo simple de transformación es la posibilidad de convertir SVG o cualquier otro documento basado en un lenguaje de etiquetas en algo diferente de forma semiautomática. Las posibilidades que se abren para la RI gráfica son obvias: conversión de texto en gráficos y combinación de metadatos y gráficos.

Figura 20. Fuente: http://www.dbxgeomatics.com/svgmapmaker/samples/

4. Conclusiones

En este artículo se ha presentado *SVG* como un medio suficientemente flexible para poder representar y gestionar información en diferentes formatos. Sobre este medio es posible desarrollar herramientas que permitan gestionar información gráfica sin que por ello se tenga que renunciar a la posibilidad de que esa información pueda ser recuperable por servicios parecidos a los utilizados en la actualidad (buscadores, directorios, metabuscadores, agentes de búsqueda, etc.).

Debido al carácter xml de *SVG* es posible combinarlo con otros vocabularios xml como por ejemplo *TVWeb*, *Smil*, *Vrml* o *Voice browser*. Se puede utilizar para representar gráficamente diversos tipos de contenido: desde directorios de edificios (ver figura 20) a información química utilizando *CML* (ver figura 3), etc. Todos los ejemplos del artículo muestran imágenes con características muy similares a las que se pueden obtener utilizando programas clásicos de edición gráfica pero, además, aportan otras ventajas: es posible generar esas imágenes en un editor xml, es posible describir esas imágenes y recuperarlas, es posible agrupar sus elementos dándoles así una estructura.

Relacionado con esto último, podemos observar en estos ejemplos que se pueden asignar metadatos a los documentos *SVG* para describirlos y, por tanto, poder recuperarlos. Si bien es cierto que existen varios formatos gráficos que permiten la inclusión de metadatos primitivos no hay que dejar de lado la mayor versatilidad que ofrecen sistemas como rdf, *Dublin core* o *XTM* para la descripción de objetos electrónicos. Las ventajas de *SVG* en este sentido es que su edición es mucho más sencilla: los metadatos se pueden incluir en los ficheros gráficos como texto plano. Además, *SVG* admite la utilización de namespaces, lo que permite insertar referencias a otros ámbitos de información. Con esto se consigue ampliar enormemente las posibilidades de descripción, ya que siempre se po-

Tabla 2

```
<?xml version="1.0" encoding="utf-8"?>
 <svg width="500" height="400">
  <rect x="50" y="30" width="400" height="270" style="fill:#E3DFCC;"/>
   <line x1="50" y1="30" x2="50" y2="300" style="stroke:#000000; stroke-width:0.1"/>
   <g style="fill:#000000; font-size:12; font-family:Arial">
    <text x="20" y="300">0</text>
   </g>
   <line x1="50" y1="300" x2="450" y2="300" style="stroke:#000000; stroke-width:0.1"/>
   <g style="fill:#000000; font-size:12; font-family:Arial">
     <text x="20" y="275">10</text>
   </g>
   <line x1="50" y1="275" x2="450" y2="275" style="stroke:#000000; stroke-width:0.1"/>
   <g style="fill:#000000; font-size:12; font-family:Arial">
    <text x="20" y="250">20</text>
   </g>
   <line x1="50" y1="250" x2="450" y2="250" style="stroke:#000000; stroke-width:0.1"/>
   <g style="fill:#000000; font-size:12; font-family:Arial">
    <text x="20" y="225">30</text>
   </g>
   <line x1="50" y1="225" x2="450" y2="225" style="stroke:#000000; stroke-width:0.1"/>
   <g style="fill:#000000; font-size:12; font-family:Arial">
    <text x="20" y="200">40</text>
   </g>
   <line x1="50" y1="200" x2="450" y2="200" style="stroke:#000000; stroke-width:0.1"/>
   <g style="fill:#000000; font-size:12; font-family:Arial">
    <text x="20" y="175">50</text>
   </g>
   <line x1="50" y1="175" x2="450" y2="175" style="stroke:#000000; stroke-width:0.1"/>
   <g style="fill:#000000; font-size:12; font-family:Arial">
    <text x="20" y="150">60</text>
   </g>
  <line x1="50" y1="150" x2="450" y2="150" style="stroke:#000000; stroke-width:0.1"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="20" y="125">70</text>
  </g>
  <line x1="50" y1="125" x2="450" y2="125" style="stroke:#000000; stroke-width:0.1"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="20" y="100">80</text>
  </g>
  <line x1="50" y1="100" x2="450" y2="100" style="stroke:#000000; stroke-width:0.1"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="20" y="75">90</text>
```

```
  </g>
  <line x1="50" y1="75" x2="450" y2="75" style="stroke:#000000; stroke-width:0.1"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="20" y="50">100</text>
  </g>
  <line x1="50" y1="50" x2="450" y2="50" style="stroke:#000000; stroke-width:0.1"/>
  <rect x="66" y="225" style="fill:red; stroke:black; stroke-width:0.5" width="16" height="75"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="66" y="310" transform="translate(66,310) rotate(90) translate(-66,-310)">enero</text>
  </g>
  <rect x="98" y="175" style="fill:red; stroke:black; stroke-width:0.5" width="16" height="125"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="98" y="310" transform="translate(98,310) rotate(90) translate(-98,-
310)">febrero</text>
  </g>
  <rect x="130" y="125" style="fill:red; stroke:black; stroke-width:0.5" width="16"
height="175"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="130" y="310" transform="translate(130,310) rotate(90) translate(-130,-
310)">marzo</text>
  </g>
  <rect x="162" y="275" style="fill:red; stroke:black; stroke-width:0.5" width="16" height="25"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="162" y="310" transform="translate(162,310) rotate(90) translate(-162,-
310)">abril</text>
  </g>
  <rect x="194" y="50" style="fill:red; stroke:black; stroke-width:0.5" width="16" height="250"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="194" y="310" transform="translate(194,310) rotate(90) translate(-194,-
310)">mayo</text>
  </g>
  <rect x="226" y="100" style="fill:red; stroke:black; stroke-width:0.5" width="16"
height="200"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="226" y="310" transform="translate(226,310) rotate(90) translate(-226,-
310)">junio</text>
  </g>
  <rect x="258" y="50" style="fill:red; stroke:black; stroke-width:0.5" width="16" height="250"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="258" y="310" transform="translate(258,310) rotate(90) translate(-258,-
310)">julio</text>
  </g>
  <rect x="290" y="150" style="fill:red; stroke:black; stroke-width:0.5" width="16"
height="150"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="290" y="310" transform="translate(290,310) rotate(90) translate(-290,-
310)">agosto</text>
  </g>
  <rect x="322" y="275" style="fill:red; stroke:black; stroke-width:0.5" width="16" height="25"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="322" y="310" transform="translate(322,310) rotate(90) translate(-322,-310)">septiem-
bre</text>
  </g>
  <rect x="354" y="50" style="fill:red; stroke:black; stroke-width:0.5" width="16" height="250"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="354" y="310" transform="translate(354,310) rotate(90) translate(-354,-310)">octu-
bre</text>
  </g>
  <rect x="386" y="75" style="fill:red; stroke:black; stroke-width:0.5" width="16" height="225"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="386" y="310" transform="translate(386,310) rotate(90) translate(-386,-310)">noviem-
bre</text>
  </g>
  <rect x="418" y="225" style="fill:red; stroke:black; stroke-width:0.5" width="16" height="75"/>
  <g style="fill:#000000; font-size:12; font-family:Arial">
   <text x="418" y="310" transform="translate(418,310) rotate(90) translate(-418,-310)">diciem-
bre</text>
  </g>
</svg>
```

drán aprovechar aquellos lenguajes creados específicamente para un área concreta.

Parece evidente, por lo tanto, que *SVG* es un medio adecuado para integrar diferentes representaciones de la información y que, además, permite acceder a ellas de una forma eficaz.

5. Notas

1. *SVG* 1.0, *SVG* 1.1 y *SVG mobile profiles* son recomendaciones del *World Wide Web Consortium*. En la actualidad se está trabajando con la versión 1.2 de *SVG*. Tiene previsto ver la luz como recomendación en enero de 2004.

2. *http://www.corbis.com*

3. *http://www.cnn.com*

4. *http://www.photos.external.lmco.com/*

5. *http://www.nasa.gov/gallery/photo/index.html*

6. *http://www.doedigitalarchive.doe.gov/*

7. *http://www.ctr.columbia.edu/VisualSEEk/*

8. Este índice puede construirse aplicando métodos diferentes: índices invertidos, índices vectoriales o lingüísticos, esquemas de bases de datos o entidades extraídas directamente del texto mediante el proceso del lenguaje natural.

9. Puede surgir cierta confusión en el artículo respecto al empleo del término imagen. Por una parte se utiliza extensivamente para referirse a los objetos de contenido gráfico en la web. En este caso se utiliza para referirse a un subconjunto de esos objetos en el que se incluyen, por ejemplo, las fotografías.

10. Con información primaria nos referimos a la información que es inherente al gráfico.

11. Un documento en formato xml no es más que un conjunto de elementos organizados jerárquicamente mediante una estructura arbórea. Un elemento puede ser prácticamente cualquier unidad de significado. Xml es un medio neutral: solamente establece las reglas sintácticas según las cuales se pueden definir "vocabularios" que posteriormente se emplearán para describir libros, términos matemáticos, cotizaciones o fotografías. Una vez definidos estos vocabularios, la tarea de automatizar la gestión de información es más fácil, porque se ejerce sobre un "modelo" abstracto de la información a tratar.

12. Un mapa de tópicos en definitiva puede ser considerado como una especie de índice.

13. Esto podría identificarse con una relación *BT* (*Broader term*) en un tesauro.

14. Se puede ver en *http://www.adobe.com*

15. *Chemical markup language*, vocabulario de la sintaxis xml especializado en términos químicos.

16. Como se verá más adelante es posible transformar documentos xml usando el lenguaje de programación *Xslt* (*Extensible stylesheet language: transformation*). En este caso se ha transformado un documento *CML* en un gráfico *SVG*.

17. Aunque esto último no es recomendable existiendo normas, como *Vrml*, específicas para gráficos en tres dimensiones.

18. *http://www.w3c.org/Graphics/SVG/*

19. Un parser es un programa que disecciona código fuente (código antes de ser compilado) para transformarlo en código objeto (lenguaje de máquina). En lingüística, el verbo inglés "to parse" significa dividir el lenguaje en pequeños componentes analizables. Analizar una oración, por ejemplo, significaría dividirla en palabras, grupos sintácticos, etc. e identificar cada uno de esos componentes. En este proceso se pueden distinguir dos partes: un análisis léxico, que divide el código en componentes básicos llamados tokens, y un análisis semántico que trata de determinar su significado.

20. No todos, por motivos de extensión.

21. Como ya se ha comentado, *SVG* implementa un gran número de propiedades *CSS*.

22. La letra mayúscula denota posición absoluta y la minúscula posición relativa.

23. Especificación del *W3C* sobre hiperenlaces

24. Esta jerarquía es una propiedad heredada de la sintaxis xml, en la cual un documento no es más que un conjunto de elementos organizados jerárquicamente mediante una estructura arbórea.

25. Los namespaces permiten que cada elemento pueda ser entendido en un entorno de información específico. Por ejemplo, los elementos xml que conciernen a las instrucciones de transformación *Xslt* están en el namespace: *http://www.w3.org/xsl/transform/1.0*. mientras que los que sólo son elementos xml de salida están en: *http://www.w3.org/xsl/format/1.0*.

26. La recomendación *Xslt* fue desarrollada por el *XSL working group* y ratificada por el *World Wide Web Consortium* en noviembre de 1999.

27. En el caso del ejemplo del apartado 2.3 el documento fuente sería el documento *CML* y el documento resultante sería el fichero *SVG*, que, interpretado por un navegador, daría lugar al gráfico del ejemplo.

28. Para operar esta transformación se ha utilizado el parser *Xslt XT* de **James Clark**.
ftp://ftp.jclark.com/pub/xml/xt-win32.zip
http://windowsupdate.microsoft.com/?IE

6. Bibliografía

Birbeck, Mark (ed.)**.** *Professional xml metadata*. 2nd ed. Wrox Press Inc., 2001. Isbn: 1861005059.

Directrices del W3C para autores de html.
http://www.w3.org/WAI/GL/

DOM (Document object model).
http://www.w3.org/DOM/
http://www.w3.org/TR/REC-DOM-Level-1/

Especificación *SVG*.
http://www.w3.org/TR/SVG/

Especificación *Xslt*.
http://www.w3c.org/TR/xslt

Gaborit, Gaëtan. *Un exemple de carte géographique interactive réalisée en SVG (Scalable Vector Graphics)*. Consultado en: 29-07-03.
http://svgmap.free.fr/carte.htm

Kay, Michael. *Xslt programmer's reference*. 2nd ed. Wrox Press Inc., 2003. Isbn: 1861005067.

Megginson, David. *Structuring xml documents*. New Jersey: Prentice Hall PTR, 1998. Isbn: 0-13-642299-3.

Oecd atlas Europa.
http://www.carto.net/papers/svg/eu/oecdatlas.html

OpenTag.
http://www. opentag.org

Página oficial del W3C para xml.
http://www.w3.org/XML/Activity

Watt, Andrew H. *Designing SVG web graphics*. New Riders Publising, 2001. ISBN: 0-7357-1166-6.

Webopaedia.
http://www.webopaedia.com

Xml (eXtensible Markup Language).
http://www.w3.org/XML/
http://www.w3.org/TR/1998/REC-xml-19980210

XSchema.
http://purl.oclc.org/NET/xschema

XSL (eXtensible Style Language)
http://www.w3.org/TR/WD-xsl/

Vital Access

The formula for enhanced access is simple: Add INSPEC® to your *ISI Web of Knowledge*℠ platform. You'll get premier coverage of physics, engineering, electronics, computing, and information technology. AND vital access to a greater quantity of highly relevant results through:

- Full integration of resources and cross-search discovery tools
- Simultaneous searching of multiple resources, with de-duplicated results
- Inter-product links, links to full-text
- Direct links to citation data in *Web of Science*®*

ISI Web of Knowledge content, tools, technology, and easy-to-use interface add new dimensions to INSPEC features users depend on:

- Global coverage to 1969 of journals, conference proceedings, books, and reports
- INSPEC Thesaurus and INSPEC Classification
- Indexing by numerical data, astronomical object data, and chemical compound data

Get vital access to the content you need most.
With INSPEC via *ISI Web of Knowledge*.

* for mutual subscribers

Produced by the IEE (Institution of Electrical Engineers)

For more information, please visit
www.isinet.com/isi/forms/inspec.

Or call +1 800 336 4474 or e-mail sales@isinet.com.

THOMSON
ISI