# Mejorando la accesibilidad de las aplicaciones GIS basadas en Web*

[1]Nieves R. Brisaboa [1]Miguel R. Luaces [1]Jose R. Parama [1]David Trillo [2]José R.R. Viqueira

*1 Laboratorio de Bases de Datos*
*Departamento de Computación*
*Universidad de A Coruña*
*Facultade de Informática,*
*Campus de Elviña, s/n. 15071. A Coruña. España*
*{ brisaboa,luaces, parama, dtrillo}@udc.es*

*2 Laboratorio de Sistemas*
*Departamento de Electrónica y Computación*
*Universidad de Santiago de Compostela*
*Instituto de Investigaciones Tecnológicas*
*Campus sur, 15782 Santiago de Compostela, España*
*joserios@usc.es*

## [1]Abstract

*El principal problema de los formatos vectoriales activos como WebCGM y Scalable Vector Graphics (SVG), radica en la necesidad de instalar plug-ins o descargar applets que permitan su visualización en la mayoría de los navegadores web. En este artículo se presenta un servicio web cuya funcionalidad permite la transformación de mapas vectoriales activos en formato SVG en una nueva representación de mapas activos, que utiliza únicamente elementos de DHTML (HTML + JavaScript), mejorando de esta forma el acceso a las aplicaciones GIS basadas en web. Esta nueva representación, que forma parte del presente trabajo, incluye una representación raster del mapa y una representación vectorial de sus objetos geográficos. La primera es utilizada como imagen de fondo del mapa, mientras que la segunda permite responder a eventos de usuario. Un índice espacial, en concreto una estructura R-Tree, es utilizada para acceder de forma eficiente a los objetos geográficos afectados por cada evento con el propósito de ejecutar las acciones correspondientes.*

## Palabras Clave

*Aplicaciones SIG basadas en Web, Lenguajes Geográficos, Representaciones Geográficas, Servicios SIG basados en Web, Interoperabilidad.*

## 1. Introducción

Los esfuerzos de investigación que se han llevado a cabo en la última década en las áreas de Sistemas de Información Geográfica (SIG) y el World Wide Web (WWW), han conducido al desarrollo de nuevas tecnologías que permiten la publicación en Web de mapas con información geográfica. En consecuencia, muchas de las aplicaciones SIG actuales son accesibles directamente a través de Internet. Un buen ejemplo de uno de estos desarrollos es la especificación Web Map Service (WMS) [4], ampliamente utilizada, propuesta por el Open Geospatial Consortium Inc. (OGC). Esta interfaz WMS, acepta una petición HTTP de un mapa y responde con el correspondiente mapa, ya sea en formato vectorial o raster.

Los formatos raster de mapas, como PNG, JPG y GIF, pueden ser insertados directamente en HTML, siendo posible su visualización en cualquier navegador. Sin embargo, la actividad de esos formatos de mapas está muy restringida. En particular, los objetos geográficos contenidos en un mapa raster no pueden responder individualmente a los eventos de usuario (movimiento de ratón y clic) con su propia funcionalidad. Esta limitación es parcialmente resuelta con la incorporación de elementos <MAP> de HTML, si bien las propie-

* Este trabajo fue parcialmente subencionado por el CICYT (refs. TIC2002-04413-C04-04 y TIC2003-06593), y la Xunta de Galicia (ref. PGIDIT02SIN10501PR).

dades visuales de los objetos no pueden ser cambiadas como respuesta a eventos de usuario. Por ejemplo, no es posible cambiar el color de relleno de un polígono como respuesta a un evento de clic sobre él.

Las limitaciones de actividad de los formatos raster, no aparecen en los formatos vectoriales activos, como WebCGM y Scalable Vector Graphics (SVG) [5] [6]. El mayor problema de estos formatos, en general, es la necesidad de un plug-in o un applet para poder visualizarlos a través de los navegadores web. Es bien conocido que el uso de estos elementos reduce considerablemente la accesibilidad de las páginas web.

En este artículo proponemos una nueva representación de mapas activos que está totalmente implementada utilizando DHTML, y cuya funcionalidad soporta las características comunes de mapas activos SVG. El código DHTML propuesto incluye una representación raster del mapa y una representación vectorial de cada uno de sus objetos geográficos implementada en JavaScript. La primera es utilizada como imagen de fondo del mapa, mientras que la segunda permite al mapa responder a eventos iniciados por el usuario. Un Controlador de Eventos implementado en JavaScript, utiliza una estructura de indexación espacial, en concreto una estructura R-Tree, para acceder a los objetos geográficos afectados por cada evento de ratón con el propósito de ejecutar la acción correspondiente (que también estará implementada en JavaScript). Finalmente, se ha desarrollado un servicio web que transforma mapas activos en formato SVG a la nueva representación DHTML.

El resto de este artículo está organizado de la siguiente forma: En la sección 2 se presenta la nueva representación DHTML para mapas activos. El controlador de eventos JavaScript se describe en la sección 3. La sección 4 está dedicada a la arquitectura del servicio web desarrollado. Las decisiones de diseño, así como las ventajas e inconvenientes de la presente aproximación, son tratados en la sección 5. Finalmente, las conclusiones y posibles trabajos futuros son identificados en la última sección.

## 2. Representación DHTML de mapas activos

En esta sección describiremos la representación DHTML de mapas activos, la cual soporta las características de SVG que son utilizadas habitualmente para modelar mapas activos. Este subconjunto de características es explicado a continuación:

*Mapas Activos en SVG*: En el contexto del presente artículo, un *Mapa Activo* es una colección de objetos geográficos (puntos, líneas y superficies), cada uno de los cuales posee las siguientes características: i) un estilo de visualización (colores del borde y del relleno, grosor de la línea, iconos, etc.) y ii) un comportamiento, expresado en algún lenguaje de tipo Script (ECMAScript, JavaScript, etc.), que permitirá al mapa responder a eventos como el movimiento o el clic del ratón. Un mapa activo en formato SVG [5] es un documento XML cuyos elementos son utilizados para describir los atributos de cada uno de los objetos geográficos que componen el mapa. Un ejemplo de mapa SVG conteniendo una superficie lo podemos ver en la Figura 1.

```
<svg viewbox="0 0 20 20" width="10px"
                         height ="10px">

  <script type="text/ECMAScript">
   <![CDATA[
    function change_colour(evt){
      var target= evt.target;
      target.setAttribute("fill","white");
    }
   ]]>
  </script>

 <g stroke="black" fill="silver"
    onclick="cambiar_color(evt)">
  <polygon points = "1 11
                     6 1
                     11 6.5
                     19 8
                     10 18">
 </g>
</svg>
```

**Figura 1**: Mapas Activos en formato SVG

El elemento <svg> es utilizado para especificar las coordenadas geográficas del *minimum bounding rectangle* (MBR), el rectángulo mínimo contene-

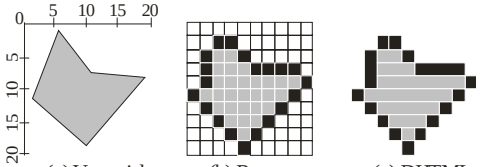
dor que engloba el mapa (viewbox en SVG), así como las dimensiones en píxeles del área de representación del mapa.

Los elementos <g> son contenedores utilizados para agrupar objetos geográficos que comparten algunas propiedades, ya sean estilos de visualización o atributos de comportamiento[2]. Los estilos de visualización son definidos como atributos de los elementos SVG (ver stroke="black" y fill="silver" en la Figura 1.). Estos atributos también son utilizados para enlazar las funciones escritas en algún lenguaje tipo Script con algún evento dado, y por lo tanto, definiendo el comportamiento de los objetos incluidos en el correspondiente elemento SVG. De esta forma, en el mapa de la Figura 1 se ha definido que cuando se produce un clic de ratón sobre algún objeto, la función *change_colour* es utilizada para rellenar su color de "blanco".

Finalmente, se ha dado soporte a varios elementos básicos de SVG, como rectángulos, círculos, polilíneas, polígonos, etc., los cuales permitirán definir las coordenadas de los elementos gráficos (ver elemento *polygon* en la Figura 1).

Para dar soporte a este tipo de mapas en la nueva representación DHTML, una funcionalidad básica requerida es el renderizado de los elementos vectoriales en HTML. Esta funcionalidad básica es obtenida de la librería *JavaScript Vector Graphics Library* implementada por Walter Zorn [7].

*Librería JavaScript Vector Graphics Library de Walter Zorn*: Esta librería contiene una colección de funciones JavaScript que permiten el renderizado de elementos gráficos (elipses, líneas, polígonos, etc.) en HTML, haciendo uso de algoritmos bien conocidos [1]. En líneas generales, podemos decir que cada píxel o conjunto de píxeles de la representación raster del objeto de entrada, es renderizada como un elemento <Div> de HTML. Para ilustrar esto, consideremos el mapa SVG definido en la Figura 1. Este mapa contiene un polígono, cuya representación geométrica vectorial se muestra en la Figura 2(a). La representación raster de este polígono es mostrada en la Figura 2(b). La figura 2(c) muestra los elementos <Div> de HTML generados cuando el polígono es renderizado con la librería JavaScript *Vector Graphics Library* desarrollada por Walter Zorn.




(a) Vectorial (b) Raster (c) DHTML

**Figura 2**: Representaciones de mapas

Renderizar todos los objetos geográficos del mapa con elementos <Div> de HTML es muy ineficiente. Para resolver este problema, utilizaremos una representación del mapa en formato raster como imagen de fondo del mapa. Las capacidades gráficas de la mencionada librería son utilizadas únicamente para cambiar las propiedades visuales de los objetos geográficos como respuesta a algún evento de usuario. Una descripción más precisa de la representación desarrollada es la siguiente:

1. *Gestor de Coordenadas*: Es un objeto JavaScript que almacena las coordenadas geográficas del MBR del mapa y las dimensiones del área en la que se representará el mapa en la aplicación cliente.
2. *Representación Raster*: Una representación raster del mapa, incluida en HTML como una imagen (jpg, png, gif, etc.).
3. *Vector de Objetos*: Un array JavaScript que contendrá un objeto JavaScript por cada objeto geográfico del mapa. Cada objeto del array contiene diferentes atributos: la representación vectorial de los objetos geográficos relevantes, sus propiedades visuales y el nombre de la

[2] Nótese que normalmente en SIG, un mapa está compuesto por capas, cada una de las cuales contiene objetos de la misma clase (ríos, carreteras, municipio, etc.) y con el mismo estilo de visualización y comportamiento.

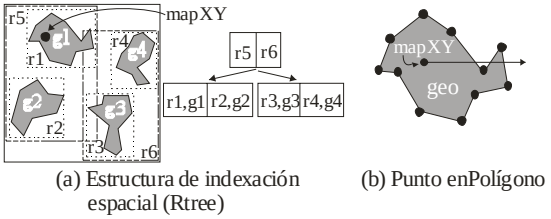
función JavaScript a ejecutar para cada posible evento de usuario.

4. *Funciones de Eventos*: El código fuente de la colección de funciones JavaScript que están referenciadas por los objetos geográficos anteriormente mencionados. Destacar que el código de estas funciones puede cambiar los atributos de los objetos geográficos anteriores. Las funciones de renderizado de la librería de Walter Zorn se utilizan para mostrar esos cambios en el navegador web.
5. *Estructuras de Indexación Espacial:* El código JavaScript de una estructura de indexación espacial, en concreto una estructura R-Tree. Esta estructura se construye a partir de la representación vectorial de los objetos geográficos incluidos en el mapa. El R-Tree es utilizado por el *Controlador de Eventos JavaScript*, el cual describiremos en la siguiente sección, para localizar de forma eficiente los objetos JavaScript afectados por cada evento de ratón, con el propósito de ejecutar la función JavaScript apropiada.

## 3. Controlador de Eventos JavaScript

El *Controlador de Eventos JavaScript* descrito en esta sección permite dar soporte a la actividad definida por las funciones JavaScript de la representación DHTML de la sección previa. Su funcionalidad se rige por el siguiente pseudocódigo.

```
(1)  while (true)
(2)  begin
(3)  event = capturaEventosDeRaton()
(4)  screenXY= obtieneLocalizaciónPuntero()
(5)  mapXY = transforma(screenXY)
(6)  result = busquedaRTree(rtree, mapXY)
(7)  for each geo in result do
(8)    begin
(9)    if contiene(geo, mapXY) then
(10)      begin
(11)      responderAEvento(geo, event)
(12)      if geo.cambiaEstilo then
(13)         pintarGeometría(geo)
(14)      end if
(15)   end for
(16)   end while
```




(a) Estructura de indexación espacial (Rtree) (b) Punto enPolígono

**Figura 3**: Obteniendo objetos geográficos afectados.

En la linea (3) el algoritmo captura un evento de ratón (mouseMove, mouseUp, MouseDown, MouseClick, etc.). Posteriormente, en las líneas (4-5), obtenemos la posición del puntero del ratón y la transformamos a las coordenadas geográficas del mapa. La estructura R-Tree disponible en la representación DHTML del mapa es utilizada en la línea (6) con el propósito de obtener una colección de objetos JavaScript que potencialmente puedan contener las coordenadas del puntero del ratón asociadas al evento. Como ejemplo, consideremos el R-Tree de la figura 3(a). En el nodo raíz, el algoritmo de búsqueda detecta que el puntero del ratón (*mapXY*) está contenido en el rectángulo r5 y continua la búsqueda por el hijo de la izquierda. Dado que *mapXY* también está contenido en r1, el objeto g1 es recuperado como resultado potencial. Para cada objeto recuperado en la búsqueda del R-Tree, el algoritmo comprueba si el puntero del ratón está realmente contenido en su geometría (línea 9). Para lograr esto, se aplica un algoritmo bien conocido de geometría computacional [3]. Un buen ejemplo de cómo uno de estos algoritmos comprueba si un punto pertenece a un polígono lo podemos ver en la Figura 3(b). Generalmente, un punto *mapXY* está contenido en un polígono *geo* si una semirrecta con origen en el punto *mapXY*, interseca con un número par de segmentos del contorno de *geo*. Si la comprobación anterior devuelve un resultado positivo entonces, en la línea (11), se invoca la función JavaScript

asociada en el objeto *geo* al evento que estamos procesando. Finalmente, si los atributos de estilo del objeto son modificados durante la ejecución de la mencionada función JavaScript, utilizaremos la funcionalidad proporcionada por las librerías JavaScript *Vector Graphics Library* de Walter Zorn para renderizar de nuevo el objeto geográfico en el navegador.

## 4. Arquitectura del servicio Web SVGTODHTML

Con el propósito de mostrar mapas SVG obtenidos de un servidor WMS estándar, ha sido desarrollado un servicio web que permite la transformación de mapas representados en formato SVG a la nueva *Representación DHTML de Mapas Activos* presentada en la Sección 2. Además, el *Controlador de Eventos JavaScript* descrito en la Sección 3 también será obtenido del servicio web. La arquitectura del servicio web SVGtoDHTML es mostrada en la figura 4.

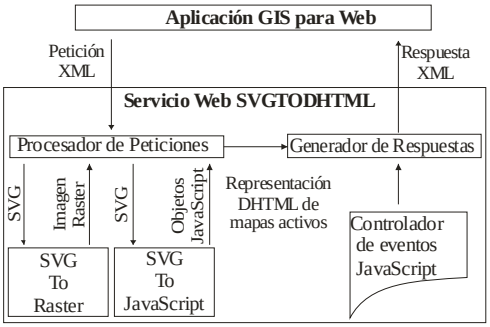



**Figura 4**: Arquitectura del servicio web SVGTODHTML

La aplicación GIS para Web accede a la funcionalidad del servicio web *SVGtoDHTML* a través de una petición POST que incluye un documento XML. El XML Schema de esta petición XML se muestra en la Figura 5(a). Como podemos ver en la figura, la petición deberá incluir un documento SVG (elemento SVG) o una referencia a una URL (elemento *url*), donde un documento SVG está listo para ser descargado[3].

Opcionalmente, la petición del servicio web SVGtoDHTML contendrá un elemento *response-Mode* que podrá incluir los siguientes elementos:

- *dhtml*: El servicio devuelve solo la *Representación DHTML de mapas activos* (ver Sección 2) generada a partir del mapa SVG incluido (elemento *svg*) o referenciado (elemento *url*) en la petición.
- *jsEventController*: El servicio devuelve solo el código del *Controlador de Eventos JavaScript* (ver Sección 3), requerida para visualizar el mapa generado en el navegador web. En este caso, la petición realizada al servicio web no incluirá un elemento *svg* o un elemento *url*.
- *all*: El servicio devuelve la *Representación DHTML de mapas activos* junto con el código del *Controlador de Eventos JavaScript.* Esta es la opción por defecto cuando el modo de la respuesta del servicio no está indicado en la petición.

El XML Schema de la respuesta XML generada por el servicio se muestra en la Figura 5(b). Esta respuesta consiste en dos elementos opcionales.

- *jsEventController*: Incluye el código fuente del *Controlador de Eventos JavaScript.*
- *dhtml*: Incluye el código fuente de la *Representación DHTML de Mapas Activos* generada. Este elemento está compuesto por otros dos elementos. El elemento *raster* incluye la imagen raster de la representación DHTML y el elemento *JavaScript* contiene el código fuente de la representación DHTML.

[3] Este elemento debe ser usado para incluir la URL de una petición *GetMap* dirigida a un servidor WMS.

```
<?xml version="1.0"?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="svgToDhtmlRequest">
  <xs:complexType>
   <xs:sequence>
     <xs:choice minOccurs="0">
      <xs:element name="svg"
                  type="xs:string"/>
      <xs:element name="url"
                  type="xs:anyURI"/>
     </xs:choice>
     <xs:element name="responseMode"
                 minOccurs="0">
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:enumeration value="dhtml"/>
        <xs:enumeration value="jsEventController"/>
        <xs:enumeration value="all"/>
       </xs:restriction>
      </xs:simpleType>
     </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

(a) Request XML Scheme

```
<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name = "svgToDhtmlResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element name = "jsEventController"
                type = "xs:string"
                minOccurs = "0"/>
    <xs:element name = "dhtml"
                minOccurs = "0">
      <xs:complexType>
       <xs:sequence>
         <xs:element name = "raster"
                     type = "xs:base64Binary"/>
         <xs:element name = "javascript"
                     type = "xs:string"/>
       </xs:sequence>
      </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

(b) ResponseXML Scheme

**Figura 5**: XML Schema de la peticiones y respuestas.

A continuación describiremos brevemente cada uno de los módulos que componen la arquitectura del servicio web SVGtoDHTML:

*Procesador de peticiones*: La petición enviada por la aplicación cliente es analizada sintacticamente y procesada por este módulo. En el caso más simple, si el modo de respuesta especificado es *jsEventController*, entonces el Generador de Respuestas se invoca para incluir solo el *Controlador de Eventos JavaScript* en la respuesta. En otro caso, la *Representación DHTML de mapas activos* debe de ser generada a partir del documento SVG de la petición. Para conseguir esto, se utiliza un módulo *SVGtoRaster* para generar la imagen raster y el módulo *SVGtoJavaScript* se utiliza para generar el código JavaScript.

*SVG To Raster*: Renderiza el documento SVG obtenido en la petición en una imagen raster. La funcionalidad requerida es proporcionada por la herramienta Open Source Batik[2].

*SVG To JavaScript*: Procesa el documento SVG obtenido por la petición recibida y genera el código JavaScript de la *Representación DHTML de mapas activos*. Este proceso puede ser resumido en los siguientes cinco pasos:

1. El código JavaScript del objeto que almacena las coordenadas geográficas del MBR del mapa y las dimensiones del área en la que será representado, se generan a partir del elemento <SVG> del documento SVG de entrada.
2. Cada función implementada en código Script y contenida en el documento SVG, se transforman en una función JavaScript, con el mismo nombre en el código JavaScript resultante.
3. Los elementos contenedores (elementos <g>) del documento SVG de entrada son procesados recursivamente para obtener una colección de objetos, cada uno de los cuales almacenará: i) las coordenadas geográficas de los objetos geográficos relevantes, ii) propiedades de visualización y iii) el nombre de la función JavaScript que será invocada para cada posible evento.
4. Todos los objetos contenidos en el documento SVG serán procesados para generar el código JavaScript del array que contendrá los objetos geográficos relevantes. Nótese que durante este proceso, las coordenadas geográficas de cada objeto deberán de ser transformadas desde

el sistema de coordenadas del mapa hacia el sistema de coordenadas correspondiente al área de representación del mapa en la aplicación cliente. Para este propósito, será necesaria la información almacenada en el objeto JavaScript generado en el paso 1.

5. Los objetos obtenidos en el paso 3 son procesados otra vez para generar el código JavaScript de la estructura R-Tree de indexación espacial.

*Generador de Respuestas*: Construye la *respuesta XML* a partir de la *Representación DHTML de mapas activos* generada por el *Procesador de Peticiones* y el código fuente del *Controlador de Eventos JavaScript.*

## 5. Discusión

En las *Representaciones Vectoriales Activas* de mapas, como webCGM[6] y SVG[5], los objetos vectoriales pueden responder a eventos de ratón invocando funciones tipo Script de la aplicación cliente que son incluidas en la representación del mapa. Las propiedades de visualización de esos objetos vectoriales también pueden ser modificadas durante la ejecución de las mencionadas funciones. Así, por ejemplo, es muy sencillo definir una función que sea invocada cada vez que se produzca un clic de ratón sobre un objeto vectorial y cuya funcionalidad cambie el color de relleno de dicho objeto y muestre alguna información alfanumérica relevante obtenida de una base de datos. El mayor problema de esas *Representaciones Vectoriales Activas*, derivado de su complejidad, es la necesidad de utilizar un plug-in o un applet para permitir su visualización en un navegador web[4]. Los plug-ins son la alternativa más eficiente para extender la funcionalidad de un navegador web. Si bien, tienen dos importantes desventajas: i) Debido a los problemas potenciales de seguridad, deben de ser instalados por un usuario que disponga de permisos de administración y ii) son dependientes de cada navegador web en particular. Por otra parte, los applets solucionan los problemas de seguridad y de dependencia con la plataforma que acarrea la utilización de plug-ins. Sin embargo, deben de ser descargados cada vez que la página web es visitada, lo que conlleva problemas de eficiencia. Además, en algunas plataformas, como Windows 2003 Server, la Máquina Virtual Java requerida para permitir la ejecución de applets no está preinstalada.

Otra aproximación opuesta en términos de funcionalidad, es la utilización de *representaciones raster* en algún formato imagen, como JPG, PNG, GIF, etc., insertada utilizando elementos <IMG> de HTML. Esta es una solución estándar que puede ser visualizada en cualquier navegador web, sin embargo, el mapa no responderá a eventos de ratón. Para incorporar cierto grado de actividad en dicha aproximación raster, es necesario asociar a la imagen un elemento <MAP> de HTML. Dicho elemento <MAP> permite la inserción de una colección de polígonos transparentes sobre la imagen. Esos polígonos responderán a eventos de ratón invocando funciones Script de la aplicación cliente. Esta aproximación es estándar y puede ser visualizada en cualquier navegador web, sin embargo, todos los objetos geográficos contenidos en el mapa son aproximados por polígonos. Además, la funcionalidad de la actividad soportada es limitada en comparación con formatos vectoriales activos como SVG. Así, por ejemplo, no es posible cambiar las propiedades visuales de los objetos geográficos del mapa como respuesta a algún evento de usuario.

La *Representación de mapas activos en DHTML* propuesta en el presente artículo aproxima la funcionalidad de una representación vectorial activa con las propiedades de accesibilidad de una representación raster. En particular, contrariamente a las representaciones vectoriales activas, la representación propuesta en el presente artículo puede ser directamente visualizada en cualquier navegador web sin la necesidad de instalar ningún plug-in ni la necesidad de descargar ningún applet. Además, a diferencia de las representaciones ras-

[4] Una excepción de esta regla general es el navegador web Mozilla, que soporta de forma nativa el formato SVG.

ter, la presentación propuesta permite la incorporación de objetos geográficos activos que responden a eventos de usuario con la ejecución de funciones Script contenidas en la aplicación cliente, posibilitando la modificación de las propiedades visuales de esos objetos.

Las desventajas de la *Representación DHTML de mapas activos*, en comparación con representaciones vectoriales activas, son las siguientes. Primero, la respuesta a eventos de ratón es mucho más rápida si la representación se visualiza utilizando plug-ins o applets. Segundo, no toda la funcionalidad del formato SVG ha sido implementada en la presente aproximación. Por ejemplo, algunas de las propiedades de estilo que pueden ser definidas para un objeto en SVG, no están soportadas por nuestra implementación. Finalmente, la presente aproximación dejará de ser útil en el momento que el formato SVG sea soportado de forma nativa por los navegadores.

Un prototipo de prueba del servicio descrito en la sección 4, ha sido completamente implementado. Finalmente, hasta donde alcanza el conocimiento de los autores, no se puede encontrar en la literatura otra *Representación de mapas activos en DHTML* con características similares a la presentada en este artículo.

## 6. Conclusiones y trabajo futuro

En este artículo se ha propuesto una nueva *Representación DHTML de mapas activos* que permite dar soporte a la funcionalidad más común definida por los mapas SVG. Esta representación puede ser visualizada en cualquier navegador web con la ayuda de un *Controlador de Eventos JavaScript*, que también es parte del presente trabajo. Por lo tanto, no es necesario instalar plug-ins ni descargar applets para acceder a las aplicaciones que utilicen esta representación para publicar mapas activos en web. También ha sido desarrollado un servicio web que permite la transformación de mapas en formato SVG a la nueva representación DHTML. Finalmente, como posibles trabajos futuros se incluyen los siguientes: Investigación de las propiedades de varias estructuras de indexación espacial que permita seleccionar aquella que mejor se ajuste a las necesidades de la aproximación propuesta. Mejorar la implementación del *Controlador de Eventos JavaScript*, de forma que se evite el renderizado en los cambios de las propiedades de un objeto en el caso de que dichos cambios retornen las propiedades de visualización del objeto a su estado inicial. Investigación sobre algoritmos de renderización de vectores para mejorar la eficiencia de las librerías *Vector Graphics Libraries* proporcionadas por Walter Zorn. De particular interés, es la minimización del número de elementos  de HTML utilizados para renderizar un vector de elementos. Extender la implementación del prototipo para incorporar todos los tipos de eventos soportados en SVG.

## Referencias


[1] Bresenham J.E., Algorithm for Computer Control of a Digital Plotter, *IBM Systems Journal*, 4(1):25-30, 1965

[2] Batik SVG Toolkit home page, retrieved March 2005 from: http://xml.apache.org/batik/

[3] Fabri A., Giezeman G.-J., Kettner L., Schirra S., Schonherr S., On the design of CGAL a computational geometry algorithms library, *Software Practice and Experience* 30:1167-1202, John Wiley & Sons, Ltd., 2000.

[4] Open Geospatial Consortium, Web Map Service OpenGIS Specification. Version 1.3, August 2004,. Retrieved March 2005 from: http://www.opengeospatial.org/specs

[5] World Wide Web Consortium, Scalable Vector Graphics (SVG) 1.1 Specification, W3C Recommendation, January 2003. Retrieved March 2005 from: http://www.w3.org/TR/SVG11/

[6] World Wide Web Consortium, WebCGM 1.0 Second Release, W3C Recommendation, December 2001. Retrieved March 2005 from: http://www.w3.org/TR/REC-WebCGM/

[7] Zorn W, Vector Graphics Library home page, retrieved March 2005 from: http://www.walterzorn.com/