Un servicio web de mapas activos: AWMS

Miguel R. Luaces (1), José R. Paramá (1), José R. Viqueira (2), David Trillo (1)

(1) Laboratorio de Bases de Datos
Universidade da Coruña
Facultade de Informática
Campus de Elviña S/N. 15071
{luaces, parama, dtrillo}@udc.es

(2) Laboratorio de Sistemas
Universidade de Santiago de Compostela
Instituto de Investigaciones Tecnológicas
Campus Sur. 15782
joserios@udc.es

Resumen

La especificación del *OpenGeospatial Consortium* (OGC) para sistemas de información geográfica más extendida define la interfaz de un servicio de mapas en web (WMS). Un servicio que implementa esta interfaz acepta una petición de una aplicación cliente y responde con un mapa codificado en un formato *raster* o en un formato vectorial como SVG (*Scalable Vector Graphics*). Sin embargo, en ambos casos, la respuesta del WMS representa un mapa estático que no puede reaccionar a las acciones del usuario. Sería de gran utilidad obtener mapas codificados en SVG activo, capaces de ejecutar acciones y cambiar dinámicamente su aspecto visual como respuesta a eventos generados por el usuario.

En este artículo se presenta la especificación de un servicio de mapas activos en web (AWMS), definido como una extensión de la especificación WMS del OGC que permite la obtención de mapas vectoriales activos codificados en el formato SVG. Dado que WMS utilizan el lenguaje estándar, propuesto por el OGC, *Styled Layer Descriptor* (SLD) para describir el conjunto de capas de información geográfica y estilos de visualización disponibles, la especificación AWMS propuesta utiliza una extensión de dicho lenguaje para describir la componente activa de los mapas. Esto se logra añadiendo un nuevo tipo de elementos SLD denominados simbolizadores activos (*Active Symbolizers*), que permiten la definición de comportamientos activos y dinámicos en los objetos geográficos que forman parte de cada capa de información geográfica.

Palabras clave: GIS, Aplicaciones SIG basadas en Web, Servicios de Mapas en Web (WMS), Información Vectorial Activa.

1 Introducción

A lo largo de los últimos años se han desarrollado diferentes métodos para obtener, analizar, procesar, y mostrar la información geográfica. La falta de estándares ha provocado que se hayan desarrollado formatos muy diferentes para representar la información dentro de los sistemas de información geográfica (SIG). En la actualidad, el *OpenGeospatial Consortium* (OGC [1]) se encarga de definir estándares libres e interoperables para el desarrollo de aplicaciones que realizan geoprocesamiento. Estas especificaciones establecen unas bases que posibilitan el desarrollo de aplicaciones de software libre y la utilización de arquitecturas basadas en servicios que ofrecen soluciones mucho más modulares. Los sistemas *OpenGIS* representan una evolución de los sistemas SIG tradicionales, en los cuales las aplicaciones monolíticas y propietarias se vuelven interoperables y extensibles.

La especificación *OpenGIS* más utilizada actualmente define la interfaz de un *Web Map Service* (WMS [2]). Un WMS recibe una petición HTTP de un cliente en la que se solicita un mapa, recupera los objetos geográficos que componen el mapa de una base de datos o un servidor de información geográfica y, de acuerdo a unas características de estilo, genera un mapa en alguno de los formatos existentes para representar información gráfica. Un WMS describe la apariencia de un mapa en términos de capas con estilo. Una capa con estilo puede ser considerada como una lámina transparente con entidades representadas simbólicamente sobre ella. Un mapa está constituido por un número concreto de capas con estilo colocadas unas sobre otras en un orden especificado. Los usuarios pueden definir mapas más sencillos o complejos añadiendo o eliminando algunas de las capas que lo componen. El estilo visual de cada una de las capas del mapa, es definido en un WMS utilizando el lenguaje *Styled Layer Descriptor* (SLD [3]). Con este lenguaje, los clientes pueden utilizar filtros para especificar que objetos geográficos son incluidos en cada capa del mapa y simbolizadores para especificar el estilo visual con el que estos objetos geográficos son renderizados en el mapa resultante.

El OGC recomienda el formato *Scalable Vector Graphics* (SVG [4]) como formato vectorial para la representación de los mapas generados por los WMS. Un mapa representado en este lenguaje, basado en XML, podrá responder a eventos de usuario y de cambiar su aspecto visual de forma dinámica. Aunque SVG soporta la descripción de actividad y dinamismo, ni WMS ni SLD consideran estas características a la hora de describir los mapas. Los servidores WMS utilizan el lenguaje SLD para definir el estilo visual de cada capa de objetos geográficos que componen el mapa; sin embargo, SLD no permite definir el comportamiento asociado a estos objetos geográficos frente a un evento de usuario determinado. Por ejemplo, si quisiésemos indicar que cuando el usuario hace “click” sobre una carretera determinada, esta cambia su color a amarillo o que debería de mostrarse una ventana con los datos de la carretera, no podríamos utilizar el lenguaje SLD.

En nuestra larga experiencia en el desarrollo de sistemas de información geográfica [6][7] hemos advertido que la actividad y el dinamismo, son características imprescindibles en los mapas que se ofrecen al usuario, sin embargo en todas las aplicaciones desarrolladas, hemos tenido que implementarla de un modo no estándar ya que la especificación WMS no soporta la inclusión de una componente activa en los mapas. Por ello, se ha decidido desarrollar el *Active Web Map Service* (AWMS): una extensión del estándar WMS que, respetando todas las características de las especificaciones OGC, amplía sus posibilidades y permite aprovechar las características de actividad de SVG.

Para seguir respetando las especificaciones del OGC, este nuevo servicio de mapas en web, AWMS mantiene la misma interfaz del WMS actual, pero acepta las definiciones de estilo en SLD estándar extendidas por medio de unos elementos denominados Simbolizadotes Activos. Este SLD

extendido, que hemos denominado SLD con *Active Symbolizer* (SLDAS) permite describir comportamientos activos asociados a cada capa de información geográfica. En este trabajo, se presenta tanto AWMS como SLDAS. El nuevo servicio AWMS acepta los estilos definidos por SLDAS, de modo que permite generar mapas codificados en SVG activo y dinámico. De esta forma se generan mapas para web provistos de una amplia funcionalidad, capacidad de respuesta frente a eventos y con los que el usuario podrá interactuar, consultar propiedades de los objetos geográficos incluidos en cada capa, etc. En la actualidad, después de haber completado la definición de SLD con *Active Symbolizer* y haber considerado el impacto de las modificaciones realizadas en el diseño del AWMS, se está finalizando con la validación empírica de las implementaciones realizadas.

En la Figura 1, se muestra la arquitectura de los WMS convencionales frente a la del AWMS que se presenta en este trabajo.

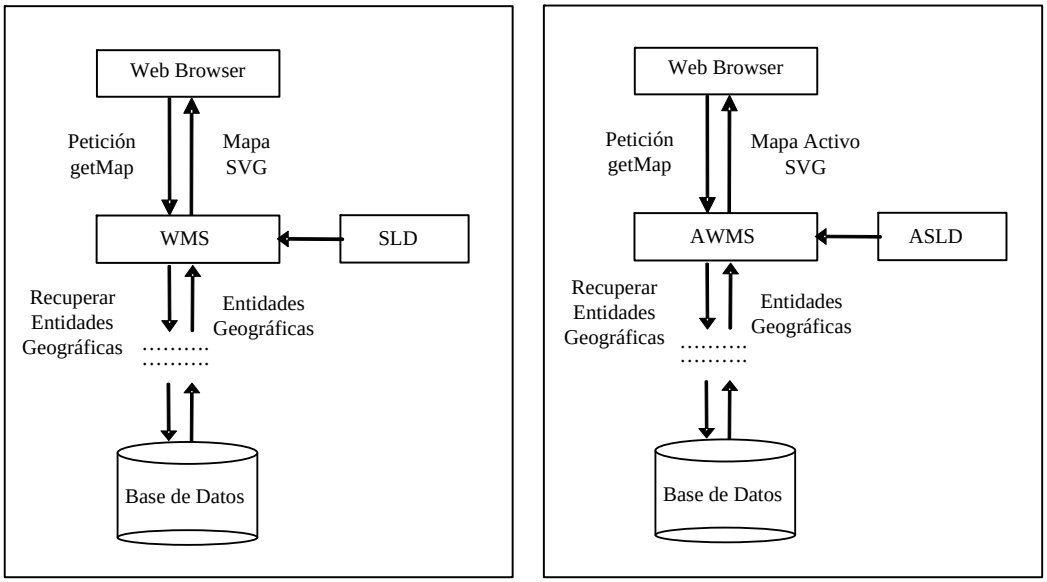

Figura 1. Un ejemplo de figura para copiar y pegar

En esta artículo se describen brevemente los estándares utilizados y se definen los conceptos más importantes implicados en el trabajo realizado, se realiza una descripción y una definición formal del concepto de mapas activos y del lenguaje SLD con Active Symbolizer. A continuación se describe el funcionamiento general del AWMS y se presentan dos aplicaciones web en las que se ha utilizado la nueva implementación de WMS. Finalmente, se describen las conclusiones que se pueden extraer de este trabajo y las posibles líneas de trabajo futuro.

2 Trabajo relacionado

2.1. Web Map Service

Un WMS, acepta peticiones HTTP de aplicaciones clientes en las que se solicita un mapa y responde con el correspondiente mapa codificado en el formato indicado en la petición y con el estilo de visualización solicitado. Las principales funcionalidades que ofrece un WMS son: (i) obtener información del servicio y de la información geográfica que éste contiene (ii) solicitar mapas en diferentes formatos y leyendas de los objetos del mapa (iii) gestionar capas de

información geográfica y estilos de visualización.

Un servidor de tipo *WMS* consta de varias operaciones, en concreto 2 obligatorias y 5 opcionales. A continuación se describen las operaciones más importantes ofrecidas por un WMS:

- *GetCapabilities (obligatoria)*: Permite obtener metadatos acerca del servicio web implementado, información de las funcionalidades soportadas por el servicio, e información específica sobre las capas de información geográfica disponibles. La información es devuelta al cliente codificada en un documento XML.

- *GetMap (obligatoria):* La funcionalidad más importante de un servicio WMS es la de generar un mapa a partir de un conjunto de entidades geográficas y de una descripción de las capas que lo componen junto con el estilo de visualización correspondiente a cada una de ellas. A través de esta operación, las aplicaciones cliente pueden obtener mapas, codificados en alguno de alguno de los formatos existentes para representar información gráfica (vectorial y raster), de forma que pueden ser visualizados directamente. Los parámetros que acepta esta operación son los que detallamos a continuación:
  - *Sistema de Referencia Espacial (SRS)* en el que se representará el mapa.
  - *Bounding Box*: Rectángulo que define la extensión geográfica del mapa en el espacio especificado por el *SRS*..
  - *Layers:* Lista de capas de información geográfica que componen el mapa.
  - *Styles:* Estilos de representación visual, descritos por un documento SLD, que se utilizarán para generar el mapa.
  - Otros parámetros, como formato de codificación utilizado, dimensiones, color de fondo, etc.

Un ejemplo de petición getMap, es el que mostramos en la Figura 2.

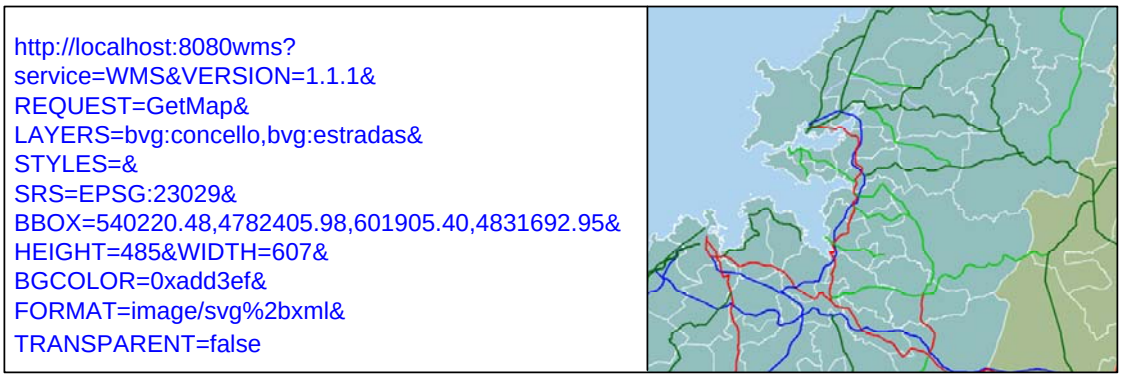

Figura 2: Ejemplo de petición getMap

2.2. Styled Layer Descriptor

Para definir las diferentes capas de información geográfica, los WMS utilizan el lenguaje *SLD*. SLD es un lenguaje basado en *XML* que codifica la apariencia que va a tener un mapa, y permite definir capas y estilos asociados a cada capa. Por ejemplo, SLD permite describir el aspecto visual de la capa de información geográfica "carreteras", expresando que éstas se representarán con un color de relleno (*fill)* de color gris, y con bordes (*stroke*, *stroke-width)* de ancho 2 y color negro.

Un servicio WMS permite gestionar los estilos de los que dispone a través de dos operaciones.

- GetStyles (opcional): Permite a las aplicaciones cliente recuperar la descripción, codificada en un documento SLD, de los estilos que se emplean para representar una capa determinada.
- PutStyles (opcional): Con esta operación, un cliente podrá definir sus propias capas y estilos, y almacenarlos en el servidor. Para realizar la inserción de un nuevo estilo en el servidor se debe incluir en la petición o bien una referencia a un documento SLD externo o bien el documento SLD completo.

Para referenciar y definir capas en un documento *SLD* tendremos elementos *NamedLaye*r que hacen referencia a las capas que ya están definidas de antemano en el servidor *WMS* y elementos *UserLayer*, los cuales hacen referencia a capas definidas por el usuario. Cualquiera de estos elementos, a su vez, puede contener elementos de tipo *NamedStyle o UserStyle*, que definen y referencian los estilos que se emplean para cada capa. Mientras que un *NamedStyle* hace referencia a un estilo ya existente en el WMS, un estilo definido por el usuario *o UserStyle* permite la creación de estilos externos a los que están definidos en el servidor.

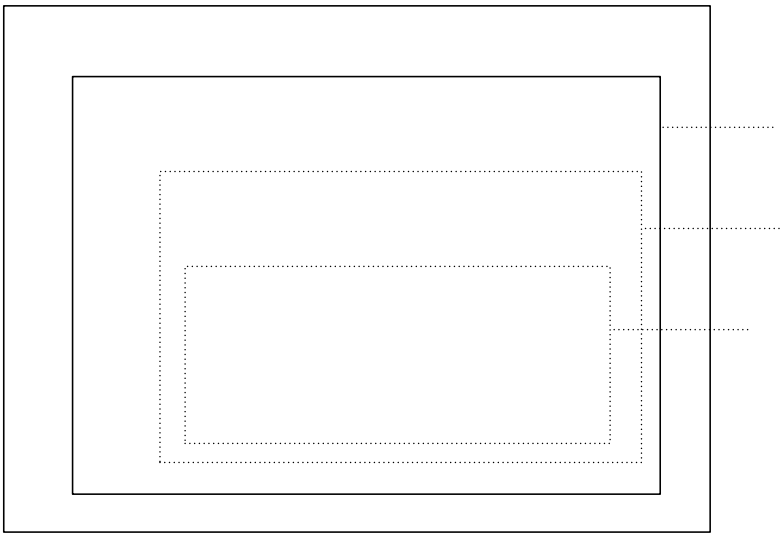

Figura 3: Ejemplo de documento SLD

Para describir los elementos geográficos de cada una de las capas, SLD dispone de elementos *FeatureTypeStyle*. Un elemento *FeatureTypeStyle* contiene elementos de tipo *Rule* para agrupar instrucciones de representación en función de las propiedades de los objetos geográficos sobre los cuales se aplica el estilo y de las escalas que se emplean para representar los mapas. Dentro de la estructura del elemento *Rule* podemos encontrar:

- Elementos *Filter o ElseFilter*, que representa un filtro acorde con las especificaciones *OGC* [5].
- Elementos *MinScaleDenominator* y *MaxScaleDenominator*, que representan los factores de escala a los cuales se puede representar la regla de un estilo.
- *Symbolizers (simbolizadores):* Describen de forma detallada el aspecto visual con el que se representarán los objetos geográficos en el mapa. Existen diferentes tipos de simbolizadotes, dependiendo del tipo de geometría que represente. Los más destacados son *LineSymbolizer*,

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0" xmlns="http://www.opengis.net/sld" ...>
<NamedLayer>
    <Name>Capa1</Name>
    <UserStyle>
        <Name>default Estilo1</Name>
        <Title>default Estilo1</Title>
        <Abstract>default Estilo1</Abstract>
        <FeatureTypeStyle>
            <Name>default Estilo1</Name>
            <Rule>
                <Name>default</Name>
                <Title>default</Title>
                <Abstract>default</Abstract>
                <MinScaleDenominator>0.0</MinScaleDenomina
                <MaxScaleDenominator>9.0E99</MaxScaleDeno
                <PolygonSymbolizer>
```

PolygonSymbolizer, PointSymbolizer, TextSymbolizer y RasterSymbolizer. Sin embargo, en el estándar actual no existen simbolizadores activos, por lo que su definición es una de las principales aportaciones de este trabajo.

En la Figura 3 se muestra un ejemplo de documento SLD, en el que se define un estilo para la capa *capa1*, dicho estilo indica que para cualquier escala de visualización, las entidades que forman dicha capa se representarán por polígonos de color gris y borde negro.

2.3. Scalable Vector Graphics

SVG es un lenguaje empleado para describir gráficos en 2D utilizando *XML*. Permite definir 3 tipos de objetos gráficos: formas geométricas vectoriales, imágenes y texto. Con *SVG* se pueden realizar gráficos animados y dinámicos, y presenta como una gran ventaja el uso de lenguajes Script, que nos proporciona acceso completo a todos los elementos, atributos y propiedades definidos en el estándar SVG. De esta forma, los gráficos dejan de ser una entidad con propiedades fijas e inmutables y pasan a poder tratarse como un conjunto de objetos con sus propios atributos y métodos.

Por otra parte el contenido de SVG puede ser interactivo. Acciones iniciadas por el usuario, como el clic o el movimiento del ratón pueden causar la ejecución de código Script o de animaciones. El formato SVG permite crear pautas de comportamiento de los gráficos (*dinamismo)* en función de los eventos generados por el usuario (*actividad).* Sin embargo ni SLD ni WMS permiten incluir actividad en los mapas, así que desaprovechan las características de actividad de SVG.

3 Mapas activos

Las características de dinamismo y actividad que nos ofrecen los documentos SVG, y que no implementan los servicios WMS actuales, posibilitan el desarrollo de mapas dinámicos, capaces de adaptar su aspecto visual a cambios de los valores que los definen y con los cuales el usuario puede interactuar.

Figura 4: Mapas Activos en formato 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:

1. Un estilo de visualización (colores del borde y del relleno, grosor de la línea, iconos, etc.)
2. Un comportamiento, expresado en algún lenguaje de tipo Script (ECMAScript, JavaScript, etc.), que permitirá al mapa responder a eventos de usuario, como el movimiento o el clic del ratón. Un mapa activo en formato SVG 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 4.

4 Active SLD

SLD utiliza simbolizadores para describir como deben de representarse las entidades geográficas que forman un mapa. Esta descripción incluye tanto la geometría que se utilizará (línea, punto, polígono) como las propiedades gráficas de esta geometría (color, opacidad, color del borde, grosor del borde, etc). Actualmente, la especificación SLD, contiene cinco tipos de simbolizadores: *Line*, *Polygon*, *Point*, *Text y Raster*, cada uno de ellos con un conjunto de atributos de estilo determinado.

Cada tipo de *Symbolizer* posee diferentes atributos que definen el estilo visual que se utilizará para representar a los objetos geográficos. Por ejemplo: (i) *Geometry* define la geometría lineal que va a ser representada. (ii) *Stroke* encapsula parámetros como *GraphicFill* y *GraphicStroke* para la representación gráfica de geometrías lineales con diferentes trazos. (iii) *Fill*, que especifica como se va a rellenar el área encerrada por los polígonos.

En un documento *SLD con Active Symbolizer*, además de poder incluir los simbolizadores definidos por el SLD Standard para definir los estilos visuales que se utilizarán para representar las entidades geográficas que componen las capas del mapa, es también posible utilizar simbolizadores activos para especificar el comportamiento asociado a cada una de esas entidades.

Un simbolizador activo (*Active Symbolizer)*, extiende el concepto tradicional de simbolizador que utiliza SLD, incluyendo atributos de actividad que determinan la función de código Script a ejecutar cuando un evento de usuario recae sobre el objeto geográfico que se está representado. De esta forma, mientras que un simbolizador tradicional incluye atributos de estilo como *fill*, *stroke*, *stroke-width*, un simbolizador activo incluirá nuevos atributos tales como *onmouseclick*, *onmouseover*, *onmouseout.* Una definición más precisa de *Active Symbolizer* utilizando XML Schema y UML es la que mostramos en la Figura 5.

Según esta definición en XML Schema, un simbolizador activo es una colección de *comportamientos activos.* Cada uno de estos comportamientos activos, está compuesto por (i) Un atributo *UserEvent*, que indica el evento de usuario al que responde este comportamiento y (ii) Una componente activa que indica la acción a realizar como respuesta al evento de usuario indicado por el atributo *UserEvent.*.

En esta definición XML Schema, se utilizarán dos elementos básicos para definir componentes activas: *ServerActiveComponent* y *ClientActiveComponent*, las cuales son descritas a continuación.

1. *ServerActiveComponent:* Se utiliza cuando el código Script que forma parte del comportamiento activo es proporcionado por el servidor. De esta forma, un elemento de tipo *ServerActiveComponent* estará compuesto por un atributo *FunctionName*, que indica el nombre de la función que se ejecutará como respuesta al evento de usuario indicado, y un

elemento de tipo *FunctionImplementation*, que incluirá la implementación en código Script de dicha función. A su vez, un elemento de tipo *ServerActiveComponent*, puede ser del tipo *NamedServerActiveComponent* o *UserServerActiveComponent*, la primera hace referencia a una componente activa existente en el servidor, mientras que la segunda, permite definir y utilizar nuevas componentes activas definidas por el usuario.

2. *ClientActiveComponent:* Este elemento se utilizará cuando el código Script que forma parte del comportamiento activo no es proporcionado por el servidor AWMS, sino que es la aplicación Web cliente la encargada de aportarlo a la respuesta que se devolverá al usuario. De esta forma, el elemento *ClientActiveComponent* estará compuesto únicamente por un atributo *FunctionName.*

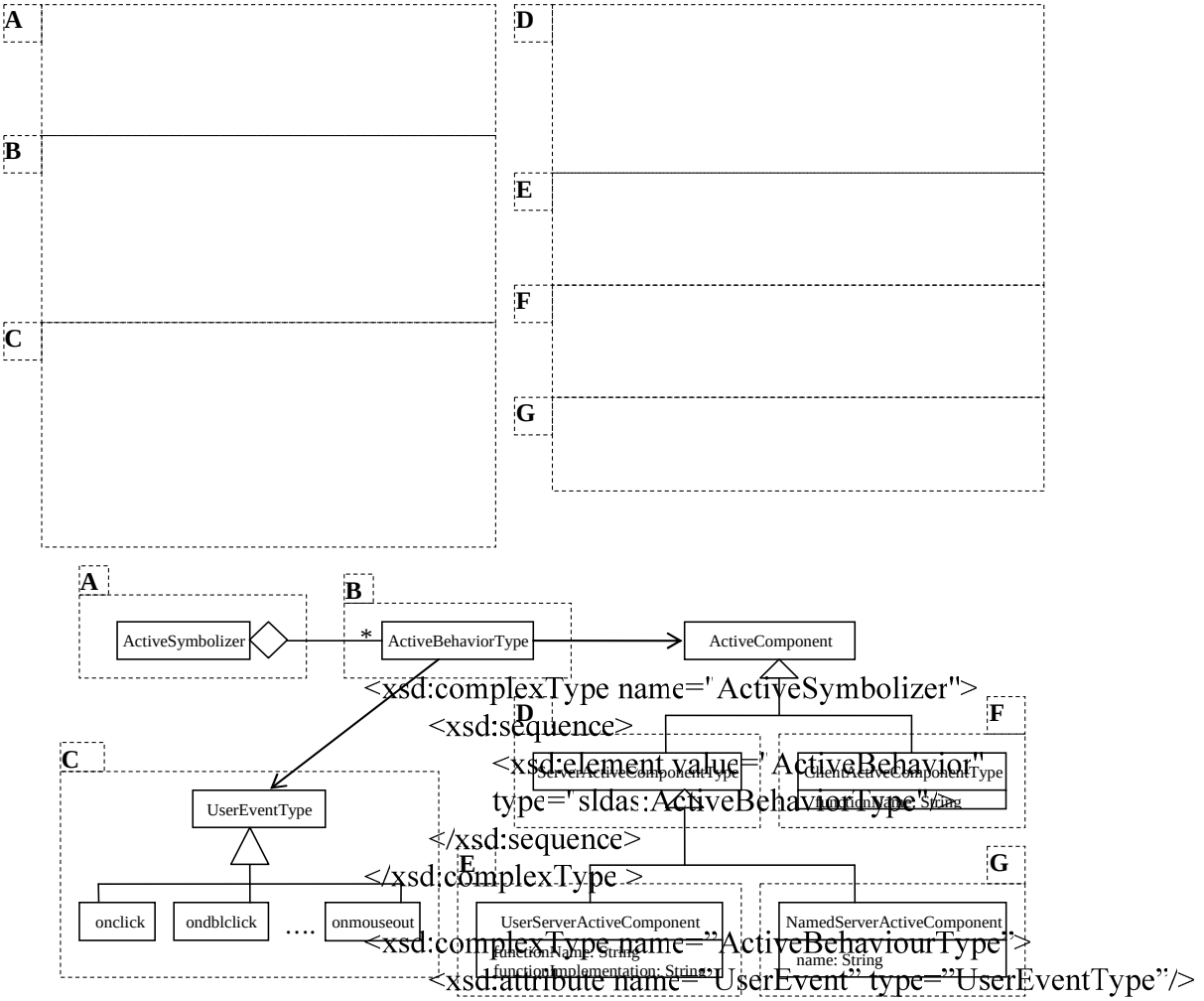

Figura 5: Descripción en XML Schema y en UML de Active Symbolizer

La principal diferencia entre la utilización de elementos de tipo *ClientActiveComponent* y *ClientActiveComponent* es el documento SVG que proporcionará el servidor AWMS como respuesta a una petición getMap. Al utilizar elementos de tipo *ServerActiveComponent*, el código Script correspondiente al comportamiento activo irá incluida en el documento SVG que codifica el mapa devuelto, mientras que al utilizar elementos de tipo *ClientActiveComponent* solo se incluirán los atributos de actividad para las entidades geográficas que componen el mapa, y será la aplicación

cliente la encargada de proporcionar la implementación de estos métodos en la página web que se proporcionará al cliente y en la que va embebido el mapa solicitado.

La utilización de elementos *ClientActiveComponent* está indicada cuando las aplicaciones cliente desean utilizar comportamientos específicos y dependientes de la implementación de la página Web en la que va embebido el mapa; por ejemplo, cuando se desea que al hacer clic sobre un objeto geográfico, se muestre su información asociada en un *TextField* determinado. Los elementos de tipo *ServerActiveComponent* son más apropiados para la definición de comportamientos activos más genéricos y no dependientes de la implementación de la aplicación cliente; por ejemplo, cuando al situar el ratón sobre un objeto geográfico, queremos que este cambie de color.

En la Figura 6 se muestra un ejemplo de simbolizador activo con componentes activas de tipo *UserServerActive component*, correspondiente al mapa SVG mostrado en la Figura 4.

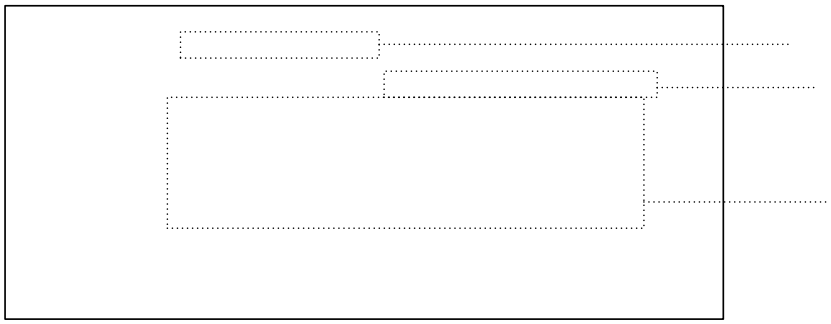

Figura 6: Ejemplo de Simbolizador Activo

Una vez que se ha descrito la extensión de la especificación SLD, SLDAS, pasamos a describir el funcionamiento del servicio AWMS frente a peticiones *getMap* de aplicaciones cliente. Podemos expresar el funcionamiento del servicio AWMS frente a estas peticiones *getMap* a partir del pseudocódigo que mostramos a continuación:

```
for each layerName-styleName in getMapRequest do                    (1)
    currentStyle = findASLDStyle(layerName, stylName)                (2)
    rules = currentStyle.getRulesForScale(currentMapScale)           (3)
    geographicObjects = getGeographicObjects(layerName)              (4)
    for each geographicObject in geographicObject do                 (5)
            for each rule in rules do                                (6)
                if evaluateFilter(geographicObject) = true then      (7)
                    for each symbolizer in rule do                   (8)
                        SVG.render(geographicObject, symbolizer)     (9)
                    end for                                          (10)
                end if                                               (11)
            end for                                                  (12)
    end for                                                          (13)
end for                                                              (14)
```

Para cada capa de información geográfica y estilo de visualización solicitado en la petición *getMap*, el algoritmo recupera la definición ASLD correspondiente (línea 2) y selecciona las reglas que corresponden con la escala actual del mapa (línea 3). Posteriormente se recuperan todos los objetos

```
<ActiveSymbolizer>
    <ActiveBehavior UserEvent="onclick">
        <ServerActiveComponent>
            <UserServerActiveComponent FunctionName="change_
                <FunctionImplementation>
                    Function change_colour(evt){
                        Var target = evt.target;
                        target.setAttribute("fill", "white");
                    }
                </FunctionImplementation>
            </UserServerActiveComponent>
        </ServerActiveComponent>
    </ActiveBehavior>
</ActiveSymbolizer>
```

geográficos correspondientes a la capa de información geográfica actual (línea 4). Cada objeto geográfico recuperado es evaluado con el objetivo de determinar si cumple alguna de las reglas definidas en el estilo de representación que se está aplicando (líneas 5, 6 y 7). Para cada una de las reglas que satisface el objeto geográfico se toma el conjunto de simbolizadores correspondiente a esa regla y se utilizan para renderizar el objeto geográfico siguiendo los estilos de visualización indicados por dicho simbolizador (líneas 8 y 9).

5 Utilización de mapas activos en aplicaciones cliente

El principal problema de los formatos vectoriales activos como SVG, radica en la necesidad de instalar plug-ins que permitan su visualización en la mayoría de los navegadores web. En esta sección presentamos dos ejemplos de aplicaciones clientes del servicio AWMS, que permiten la visualización de los mapas activos en los navegadores web, sin la necesidad de instalar plugins. La primera alternativa está basada en DHTML (HTML + JavaScript) mientras que la segunda, en Applets Java. Estas aplicaciones mejoran notablemente el acceso a las aplicaciones GIS basadas en Web, ya que facilitan la visualización de mapas activos en los navegadores web, sin la necesidad de instalar ningún componente adicional.

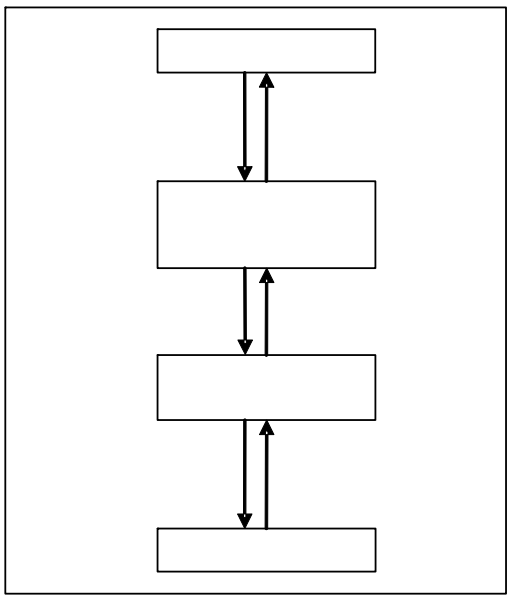

Figura 7.1: Aplicación Cliente basada en DHTML

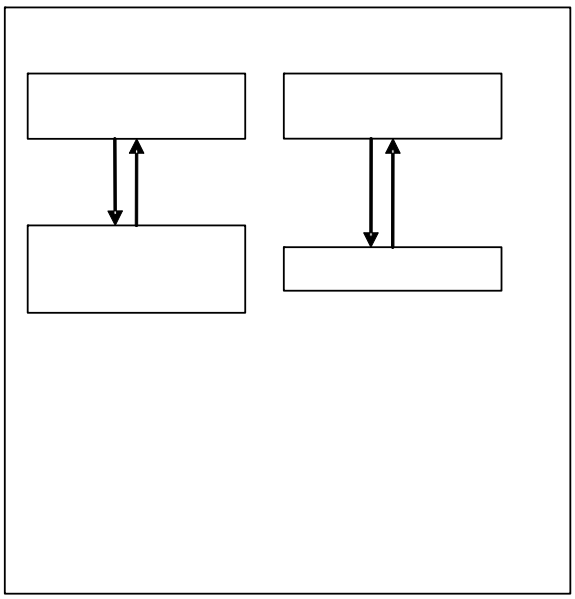

Figura 7.2: Aplicación Cliente basada en Applets Java

5.1.- Aplicación Cliente basada en DHTML.

La primera aproximación está compuesta por un Servicio Web (*SVGtoDHTML)* [6], que permite a las aplicaciones GIS basadas en Web transformar los 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 los mapas proporcionados por el servidor AWMS. Esta nueva representación DHTML, que ha sido denominada como *Representación de mapas activos en DHTML* , incluye una representación *raster* del mapa y una representación

Web Browser

Petición get
(contiene URL getMap)

Respuesta DHTML
con Mapa Activo

P

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 y cambiar de forma dinámica el aspecto visual del mapa.

Las aplicaciones GIS para Web acceden a la funcionalidad del servicio web *SVGtoDHTML* a través de una petición XML *post* que incluye un documento SVG activo o una URL que contiene una petición *getMap* al servicio AWMS. El servicio *SVGtoDHTML* transforma el mapa codificado en SVG en su representación DHTML equivalente, que podrá ser visualizada directamente por el navegador web del usuario.

La arquitectura de esta aproximación se muestra en la Figura 7.1.

5.2.- Aplicación Cliente basada en Applets Java

Mientras que en la anterior aproximación se utilizaba un servicio web para transformar los mapas activos de SVG a una representación DHTML equivalente, en este caso, la aplicación web proporciona al navegador web del usuario un applet Java que permite visualizar y ejecutar mapas activos en el ordenador del usuario.

En este caso, cuando el usuario accede por primera vez a la aplicación, la aplicación GIS basada en Web, proporciona al navegador del usuario un applet capaz de renderizar y ejecutar documentos SVG. A partir de este momento, cada vez que el usuario solicite un nuevo mapa, será el propio applet de usuario el que realice la petición al servidor AWMS.

La arquitectura de esta aproximación se muestra en la Figura 7.2.

6 Representación de mapas activos utilizando DHTML

La representación de mapas activos utilizando DHTML se basa en dos componentes principales:

1. Una representación vectorial activa para mapas geográficos implementada mediante una combinación de DHTML y JavaScript. La representación incluye un componente JavaScript que permite gestionar un modelo de objetos geográficos equivalente al del lenguaje SVG. El componente desarrollado integra el código Script incluido en los documentos SVG de entrada, y lo utiliza para responder a los eventos de usuario que afecten a los objetos geográficos representados en el mapa. Para gestionar los eventos de ratón, se utilizan estructuras de indexación espacial que permiten localizar de forma rápida los objetos geográficos afectados en función de las coordenadas del puntero. Los objetos geográficos, en respuesta a los eventos, podrán actualizar su aspecto gráfico en la página Web utilizando las librerías de gráficos vectoriales JavaScript Graphics, de forma que los gráficos generados son dinámicos.

2. Un servicio Web que permite a las aplicaciones Web obtener representaciones en DHTML y JavaScript a partir de documentos SVG. El servicio Web es capaz de extraer todos los elementos básicos como geometrías, imágenes y texto, contenidos en el documento SVG que reciba de entrada y generar, a partir de ellos, la instancia del componente JavaScript apropiada. Los objetos geográficos generados por el servicio continenen las propiedades básicas de estilo y actividad definidas en el documento SVG.

6.1 Representación vectorial activa DHTML

La representación DHTML de Mapas Activos desarrollada está compuesta por los siguientes elementos:

- *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.
- *Representación Raster*: Una representación raster del mapa, incluida en HTML como una imagen (jpg, png, gif, etc.).
- *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 función JavaScript a ejecutar para cada posible evento de usuario.
- *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.
- *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.
- *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.

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

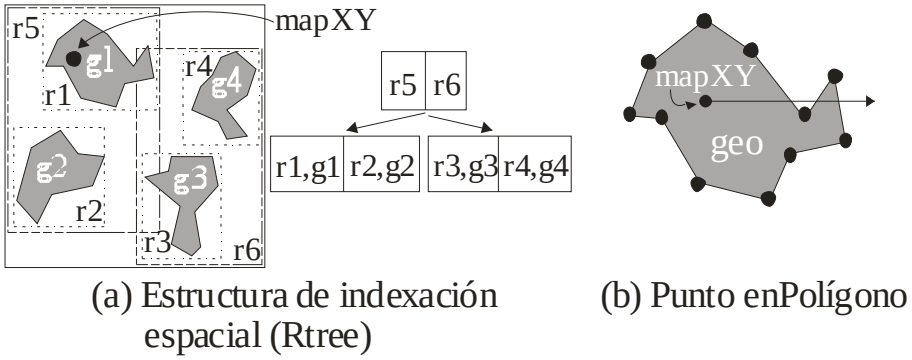

Figura 8: Obteniendo objetos geográficos afectados.

A continuación realizamos una descripción paso a paso del pseudocódico del *Controlador de Eventos JavaScript:*

1. En la linea (3) el algoritmo captura un evento de ratón (mouseMove, mouseUp, MouseDown, MouseClick, etc.).
2. Posteriormete, en las líneas (4-5), obtenemos la posición del puntero del ratón y la transformamos a las coordenadas geográficas del mapa.
3. 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 8(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.
4. 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 [8]. Un buen ejemplo de cómo uno de estos algoritmos comprueba si un punto pertenece a un polígono lo podemos ver en la Figura 8(b). Generalmente, un punto mapXY está contenido en un polígono geo si una semirrecta con origen en el punto mapXY, interseca con un número impar de segmentos del contorno de geo.
5. 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.
6. 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.

6.2. Servicio SVGtoDHTML

El servicio web *SVGtoDHTML* permite la transformación de mapas representados en formato SVG a la nueva *Representación vectorial activa DHTML*. La arquitectura del servicio web SVGtoDHTML es mostrada en la figura 9.

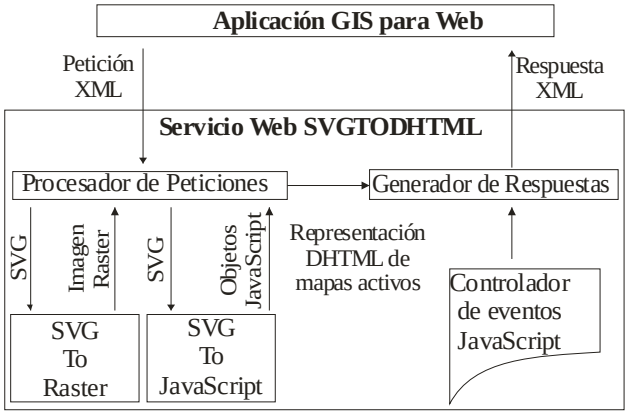

Figura 9: Arquitectura del servicio web SVGTODHTML

La aplicación GIS para Web accede a la funcionalidad del servicio web *SVGtoDHTML* a través de una petición POST que incluye un documento XML. La petición deberá incluir un documento SVG o una referencia a una URL, donde un documento SVG está listo para ser descargado[1].

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[9].
- *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,

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

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.

7 Conclusiones y trabajo futuro

En las *Representaciones Vectoriales Activas* de mapas, como SVG, los objetos vectoriales pueden responder a eventos de ratón invocando funciones de 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 asociada a dicho objeto.

En este artículo se ha propuesto una extensión de la especificación WMS que permite la obtención de mapas vectoriales activos codificados en formato SVG. Esta nueva especificación, que ha sido denominada AWMS, utiliza una extensión del lenguaje SLD *SLD con Active Symbolizer* en la que se utilizan simbolizadotes activos para describir los componentes interactivos y dinámicos de las entidades geográficas que forman parte de las capas del mapa. También ha sido desarrollada una implementación de la especificación AWMS la cual permite a las aplicaciones GIS basadas en Web obtener mapas codificados en SVG activo.

Un problema común de las *Representaciones Vectoriales Activas*, derivado de su complejidad, es la necesidad de utilizar un plug-in para permitir su visualización en un navegador web[2]. 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.

Como muestra de aplicaciones cliente del servicio AWMS desarrollado, se han presentado dos aplicaciones GIS basadas en Web, que utilizan el servicio para obtener mapas activos codificados en SVG, y los muestran al usuario utilizando diferentes técnicas de representación alternativas a los plugins: DHTML con JavaScript o Applets Java.

La aproximación basada en la *Representación de mapas activos en DHTML* proporciona la funcionalidad de una representación vectorial activa con las propiedades de accesibilidad de una

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

representación *raster*. La otra aproximación, la aplicación GIS basada en Web que utiliza un applet para visualizar mapas activos codificados en SVG, proporciona un mejor rendimiento a la hora de renderizar los objetos geográficos que componen el mapa, y una mayor funcionalidad debido a las facilidades que proporciona Java frente a JavaScript, si bien es necesario que el ordenador del usuario tenga instalada la máquina virtual de Java para poder ejecutar esta aplicación.

Finalmente, como trabajo futuro a realizar, se incluye la formalización de la especificación de AWMS para incorporar a las especificaciones OpenGIS, fomentando de esta forma el uso estándar de los servicios de mapas activos en Web.

Referencias

[1] Open Geospatial Consortium. Open Geospatial Consortium Specifications. Retrieved March 2006 from: http://www.opengeospatial.org

[2] Open Geospatial Consortium. Web Map Service Specification. Version 1.3. August 2004. Retrieved March 2006 from: http://www.opengeospatial.org/specs

[3] Open Geospatial Consortium. Styled Layer Descriptor. Version 1.0.0. September 2002. Retrieved March 2006 from: http://www.opengeospatial.org/docs/

[4] World Wide Web Consortium. Scalable Vector Graphics (SVG) 1.1 Specification. January 2003. Retrieved March 2006 from: http://www.w3.org/TR/SVG11/

[5] Open Geospatial Consortium. Filter Encoding. Version 1.0.0. September 2001. Retrieved March 2006 from: http://www.opengis.org/docs/

[6] Nieves R. Brisaboa, Miguel R. Luaces, José R. Paramá, David Trillo, Jose R. R. Viqueira. Improving Accessibility of Web-Based GIS Applications. In Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA 2005). Donald F. Shafer (Ed.), pp. 490-494. Copenhagen (Denmark) 2005. IEEE Computer Society.

[7] Brisaboa, N. R., Fariña, A., Luaces, M. R., Paramá, J. R., Penabad, M. R., Places, A. S., Viqueira, J. R. Using Geographical Information Systems to Browse Touristic Information. IT&T: Information, Tourism and Technology, vol. 6, num. 1, pp. 31-46. USA, 2003.

[8] 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.

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