# Definition and Implementation of an Active Web Map Service

Nieves R. Brisaboa [1], Antonio Fariña [1], Miguel R. Luaces [1], David Trillo [1], José R.Viqueira [2]

[1] Database Laboratory
Facultade de Informática,
University of A Coruña
Campus de Elviña, 15071. A Coruña. Spain
{brisaboa, fari, luaces, dtrillo}@udc.es

[2] Systems Laboratory
Department of Electronics and Computer Science,
University of Santiago de Compostela
Instituto de Investigaciones Tecnológicas
Campus sur, 15782 Santiago de Compostela, Spain
joserios@usc.es

**Abstract.** The most widespread Open Geospatial Consortium (OGC) specification for GIS defines the interface of a Web Map Service (WMS). A service implementing this interface accepts an HTTP request from a client and replies with a map encoded in either a raster format or a vector format such as SVG (Scalable Vector Graphics). However, in both cases, the response of the WMS represents a static map that cannot react to user actions. It would be useful to obtain vector maps encoded in active SVG that can execute actions and change their visual appearance in response to user-triggered events.

In this paper, we present the specification of an Active Web Map Service (AWMS), which is defined as an extension of the OGC WMS specification that allows the retrieval of active SVG maps. Given that a WMS uses the OGC SLD language (Styled Layer Descriptor) to describe the set of layers and visualization styles available, our AWMS specification also needs an extension of such specification to describe the active component of the maps. This is achieved by adding a new type of SLD element called Active Symbolizer that enables the definition of active and dynamic behaviour for the geographic objects that belong to each geographic information layer.

**Keywords:** GIS, web services, Web Map Service (WMS), Active Vectorial Information

## 1 Introduction

Many different methods have been defined in the last years for querying, analyzing, processing and visualizing geographic information. The lack of standards caused that many different data formats have been defined to represent information in geographic information systems (GIS). Currently,

This work was partially supported by Xunta de Galicia (refs. PGIDIT05SIN10502PR and 2006/4) and Ministerio de Educación y Ciencia (refs. TIC2003-06593 and TIN2006-15071-C03-03)

the Open Geospatial Consortium (OGC) [1] is defining free and interoperable standard specifications for the development of geoprocessing applications (i.e., GIS applications). These specifications set the ground that enables the development of open source service-based applications with a highly modular architecture. Applications based on the OGC standards improve traditional GIS applications, which are based on monolithic and proprietary architectures, by enabling the development of interoperable and extensible applications.

The most widespread OGC specification for GIS defines the interface of a *Web Map Service* (WMS) [2]. A service implementing this interface accepts an HTTP request from a client and retrieves the geographic objects that form the map from a database system, a geographic information service, or any data source. Then, the answer map is composed by applying to each geographic object a visualization style and encoding the result in either a raster or a vector image format that is sent back to the client.

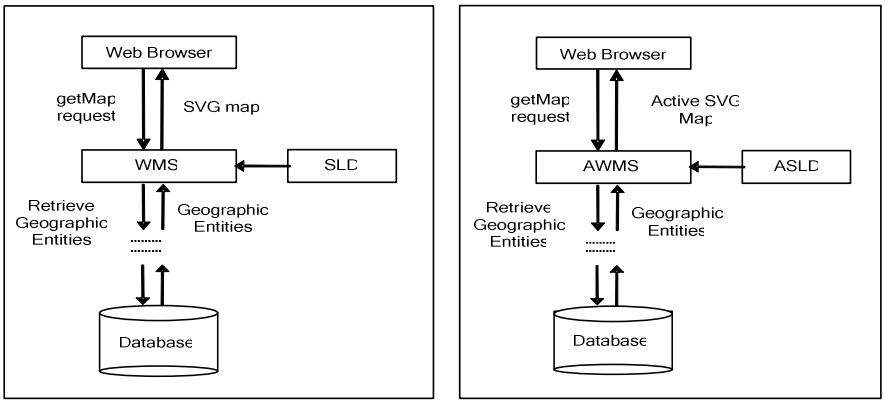
Information in a WMS is structured in *layers*. Each layer can be seen as a transparent sheet with a set of geographic objects represented on it. A *map* consists of an ordered stack of layers so that geographic objects in the top layers hide those in the bottom ones. Clients of the service can request maps of any complexity by adding or removing layers from the request. Visualization styles for each layer in a WMS are defined using the SLD language (*Styled Layer Descriptor*) [3]. With this language, clients can define which geographic objects are included in each layer by means of a filter and how the geographic objects are rendered in the resulting map. The OGC recommends the SVG language (*Scalable Vector Graphics*) [4] to encode the maps generated by a WMS. This XML-based language defines a vector graphic format that can include user-defined scripts to react to user events and to dynamically modify the visual representation of the geographic objects displayed.

Even though SVG includes support for user-defined activity and dynamic responses to user events, neither the WMS nor the SLD specifications take this functionality into account. That is, SLD does not allow the association of any behavior to geographic objects as a reaction to a given user event. For instance, we could not use the SLD language if we wanted to define a map where road sections changed their color when the user moved the mouse over them, or a map where an information window appeared when the mouse was clicked on a road section.

Due to our experience in the development of GIS applications [7, 8], we know that activity and dynamic responses to user events are required in any user interface for a GIS application. However, we always had to implement this functionality in a non-standard way because it was not included in the WMS or the SLD specifications. Thus, we propose in this

paper an extension of the WMS standard (called *Active Web Map Service* or *AWMS* for short) that maintains all the characteristics of this standard while enabling clients to define activity and dynamic responses to user events for each layer. This activity and dynamic responses will be supported by the advanced capabilities of SVG or any other format with the same capabilities.

In order to be able to define visualization styles for layers that associate behavior to the geographic objects in the map, the language that is used to define the styles also has to be extended. This is why we have also extended the specification of the SLD language by means of a new element type called *Active Symbolizer*. We have called the resulting language *Active SLD*, or *ASLD* for short. This new language also conforms to the SLD standard in the sense that any SLD file is also an ASLD file.




a) Client of WMS Service b) Client of AWMS Service

**Fig. 1.** Architecture of Web Applications

Figure 1 depicts the architecture of a web application that acts as a client to this kind of services. In Figure 1(a), the web application is a client of a WMS service. Maps are requested invoking the *getMap* operation of the service. Then, the service retrieves the geographic objects from the data source (a DBMS in this case) and uses the visualization styles defined using the SLD language to render the geographic objects in a map, which can be encoded using any image data format (e.g. SVG). In Figure 1(b), the web application acts as a client of an AWMS. In this case, maps are requested using the same operation and the service retrieves the geographic objects in a similar manner. The difference appears when the information is rendered because visualization styles can be defined using the ASLD language that enables clients to associate scripts to user events. The result-

ing map is active in the sense that the viewer can interact with the objects displayed in the map.

We have already completed the definition of the ASLD language, and we have analyzed the effect of this new language in the WMS specification that led us to the definition of the AWMS specification. We are currently working on the verification of a prototype implementation of this service.

The rest of the paper is structured as follows. Section 2 presents some related work and describes some basic concepts needed to understand the paper, particularly some of the OGC standards already introduced. The formal definition of active maps is presented in Section 3, whereas Section 4 defines the ASLD language. Section 5 describes the general processing algorithm of a service implementing the AWMS specification. Then, in Section 6 we describe two web applications that were implemented using our prototype AWMS implementation. Finally, the last section of the paper presents some concluding remarks and future lines of work.

## 2 Related Work

A WMS accepts map requests from client applications using the HTTP protocol and replies with the map encoded in the format specified in the request. The visualization style that has to be used to render each geographic object is also specified in the request and defined using the SLD language. The main functionality offered by a WMS includes: i) obtaining metadata of the service (e.g., list of layers offered, list of styles available), ii) rendering maps composed of sets of layers each with a specific visualization style, and iii) acting as a visualization style library allowing users to manage the layers and visualization styles available in the service.

This functionality is offered by implementing a number of operations, particularly two mandatory and five optional ones. We describe now briefly some of these operations:

- *GetCapabilities*: This operation replies with metadata of the service. Particularly, a client application can discover the functionality supported by the service, the list of layers and styles available, and detailed information of each layer or style.
- *GetMap*: This operation represents the main functionality of the service. Figure 2 shows an example *getMap* request. The first line is the URL of the service whereas the rest of the lines are parameters of the request. The parameter LAYERS is used to specify the layers that compose the map. In our example, two layers are requested: *bvg:municipalities* and

*bvg:roads*. Similarly, the parameter STYLES is used to indicate the visualization style that has to be applied to each layer. In our example it has been left blank and therefore the default style is applied. The portion of the world that has to be displayed in the map is specified using the parameters SRS and BBOX. The first one represents the spatial reference system to be used in the map, and the second the bounding box of the world that we want to display in the map (in coordinates of the spatial reference system). Finally, the properties of the resulting map image are specified in the remaining parameters of the request, particularly the representation format (SVG in this case).

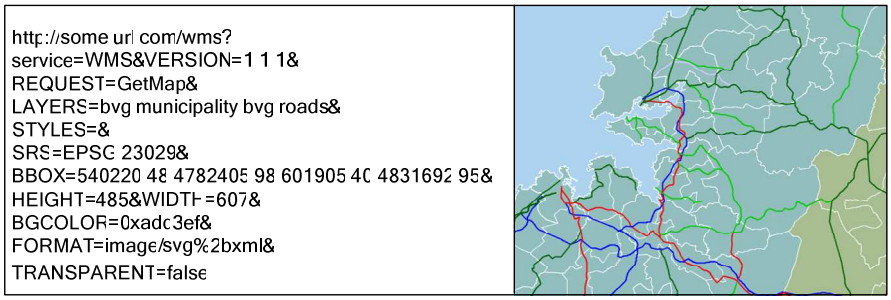



**Fig. 2.** Example of a *getMap* request

There are two types of layers and styles: those predefined by the service and those defined by clients of the service. The WMS specification provides operations that enable clients to use a WMS as a library of styles. Using these operations a client can add and remove the definitions of layers and styles. The operations are the following two:

- *GetStyles*: It enables a client to retrieve the definition of any style represented using the SLD language.
- *PutStyles*: Using this operation, a client can add or remove style definitions from the service. The styles are defined using SLD.

Finally, an additionally parameter of a *getMap* request, named SLD_BODY, enables a client to request a layer that is not predefined in the service. The definition of this layer must be given using SLD. SLD (Styled Layer Descriptor) is an XML-based language to encode the visual appearance of the geographic objects in a layer. For instance, SLD can be used to specify that roads in a map must be rendered as two pixel wide red lines. Two different types of layers can be defined in a SLD document: *named layers* and *user layers*. The former represents layers whose definition is known by the service (for instance, because it is defined in a configuration file), whereas the latter represents user-defined layers. Both

types of layers are associated to a number of styles, which can be in turn *named styles* or *user styles*. The meaning of each type is similar to that of layers: named styles are predefined in the service whereas user styles are defined by a client.

Each user style consists of a collection of rules. Each rule defines the way a particular set of geographic objects is rendered and it is composed of the following elements:

- *Filter*: The specification of the geographic objects to which this rule has to be applied. The filter is represented using the language defined by the OGC in [5].
- *Scale range*: The minimum and maximum scales at which the geographic objects are rendered. If the current scale of the map being rendered is not between these limits, the geographic objects selected by the rule filter are not rendered in the result map.
- *Symbolizers*: For each rule, a set of symbolizers can be given. Five different types of symbolizers are defined in the SLD specification: a *point symbolizer* that renders the object as a symbol, a *line symbolizer* that renders the geographic object as a line, a *polygon symbolizer* that renders the geographic object as a filled polygon, a *text symbolizer* that renders the geographic object as a text label, and a *raster symbolizer* that is used to render raster images.

The fact that a style is composed of a set of rules enables the user to render different sets of geographic objects in a layer with different visualization styles and hence compose a thematic map. For example, the style definition for a layer that renders municipalities with more than 50000 inhabitants in dark blue and the rest in light blue consists of a style with two rules. The first one selects the municipalities with more than 50000 inhabitants and associates a polygon symbolizer to them that renders the geographic objects with a dark blue fill. The second one selects the rest of the municipalities and associates a polygon symbolizer to them that renders the geographic objects with a light blue fill.

Similarly, the fact that a rule can have a set of symbolizers enables a user to render a single geographic object with different simultaneous visual representations. For instance, the style definition of a layer that renders municipalities with a light yellow fill and a label with its name consists of a rule with two symbolizers: a polygon symbolizer to render the municipality surface and a text symbolizer to render the text label with the name.

Figure 3 shows an example SLD document that defines a new visualization style for the layer *bvg:roads* that renders the geographic objects of this layer as polygons with a black border and grey fill.

After the WMS retrieves the geographic objects and renders them using the visualization style defined in the SLD document, the resulting map has to be encoded in the format specified by the user in the request. One of the most common two dimensional vector graphics formats is the SVG language (Scalable Vector Graphics). SVG is an XML-based language that supports three different types of graphic objects: vector geometric shapes, images, and text. SVG can also be used to represent dynamic animated graphics. One of its main advantages is that script languages can be used to describe the response to user-triggered events. These script functions have complete access to all the properties of the graphic objects, and hence a script can modify any attribute of any object such as the fill colour, the line width, or the text position.

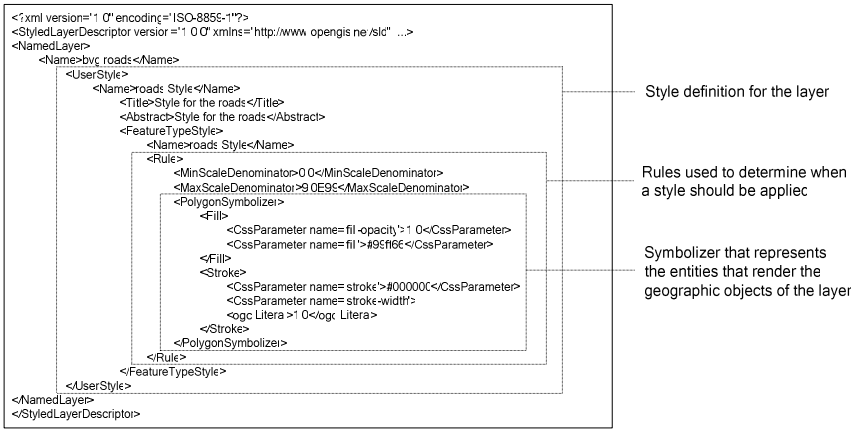



**Fig. 3.** SLD Document Example

To the best of our knowledge, there have been few attempts to extend SLD to include new funcionality. In [6], the authors propose such an extension by extending SLD with characteristics of the W3C Cascading Style Sheets and by observing the demands of location based services. However, the paper only describes motivation and intent and there is no reference to related work, the model is quite simple, and there is no validation of the proposal with real-world applications.

## 3 Active maps

Even though SVG supports dynamic graphics, neither the WMS standard nor the SLD language allows clients to define visualization styles for layers that include dynamic behaviour. In this paper, we make use of the ad-

vanced characteristics of the SVG language to define a new representation for maps that includes dynamic behaviour of its geographic objects.

We define an *active map* as a collection of geographic objects (points, lines and surfaces), each of them having the following characteristics:

- *A visualization style*: border and fill colours, border width, symbol associations, etc.
- *A behavior*: expressed using a script language (such as ECMAScript or JavaScript).

An active map can be represented using the SVG language or any other language that supports these characteristics. Figure 4 shows an example of an active map represented in SVG. In this case, the map consists of a single geographic object (a silver polygon) whose colour changes to white when the user clicks the mouse on it. This is achieved by associating the script function *change_colour* to the geographic object.

```
<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=" change_colour (evt)">
        <polygon points=" 1 11 6 1 11 6 5 19 8 10 18">
    </g>
</svg>
```

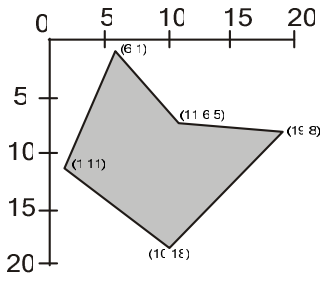



**Fig. 4.** Active Maps in SVG Format

## 4 Active SLD

As we have already seen, the SLD language uses the concept of *symbolizer* to define the visual representation of geographic objects. The specification defines five types of symbolizers and each consists of a collection of style attributes. By giving values to these attributes a user can define a specific visualization style. For instance, a polygon symbolizer allows a user to set the fill colour as well as the fill pattern. Similarly, a text symbolizer defines attributes for the font family or the font size.

In order to enable the definition of visualization styles that associate a given behaviour with geographic objects without modifying the SLD standard, a new type of symbolizer has to be defined. An *active symbolizer* represents the behaviour associated to a set of geographic objects by means

of a collection of attributes that specify the event that triggers the behaviour and the script function that implements the response to the event.

Figure 5 shows the definition of the content of an active symbolizer. Two different languages are used for this definition: an XML Schema specification and a UML class diagram. Corresponding parts of the definition are enclosed in dashed boxes and labelled with the same letter.

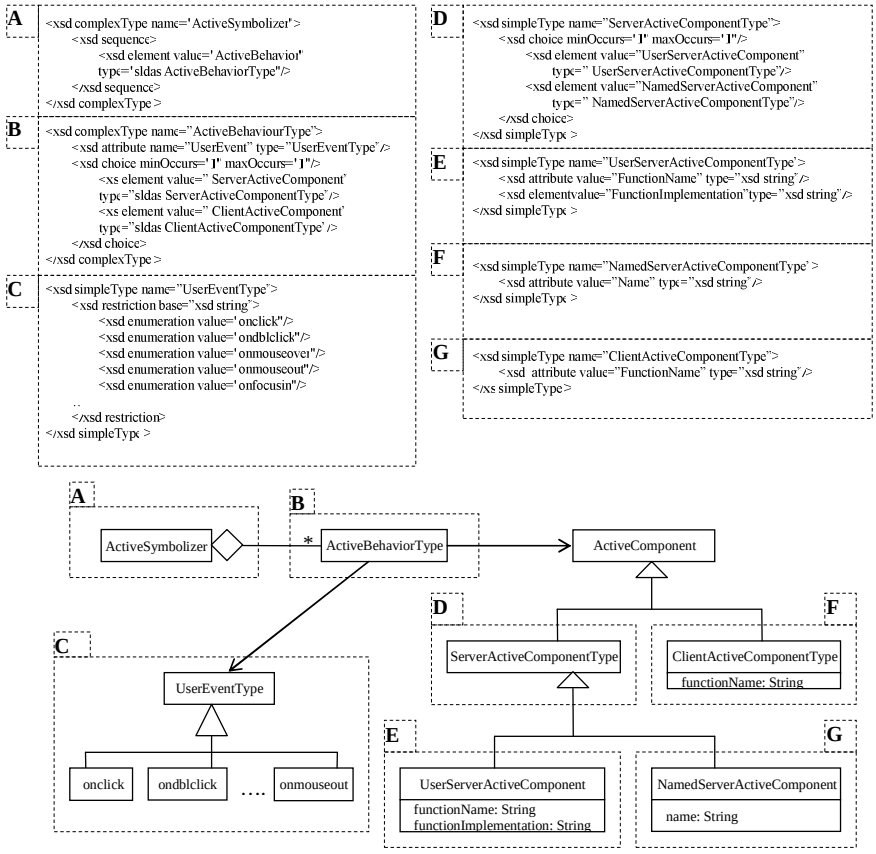



**Fig. 5.** Active Symbolizer Definition using XML Schema and UML

An active symbolizer is composed of a collection of *active behaviors.* Each of them is in turn composed of a *user event* and an *active componen*t. The former specifies the particular user event to which the behavior is associated, and the latter specifies the action that will be executed when the event is triggered. The fact that an active symbolizer consists of a set of active behaviors enables a user to specify in a single symbolizer different be-

haviors in response to different user events on the geographic object. For instance, a single symbolizer can be used to specify the behavior for mouse clicks and mouse moves.

An active component can be of two different types: a *server active component* or a *client active component*. The difference between them is that the script code of a server active component is provided by the service, whereas in a client active component the script is provided by the application requesting the active map. Server active components enable client applications to receive self-contained active maps, in the sense that no external scripts are needed to render them. This type of active components is more suitable when the behavior is generic and can be reused across many applications, such as highlighting a geographic object. On the other hand, applications using client active components can use scripts in the active maps that are not known to the service. This is useful when the client application needs a very specific behavior of the geographic objects, such as invoking a particular functionality available on the client side.

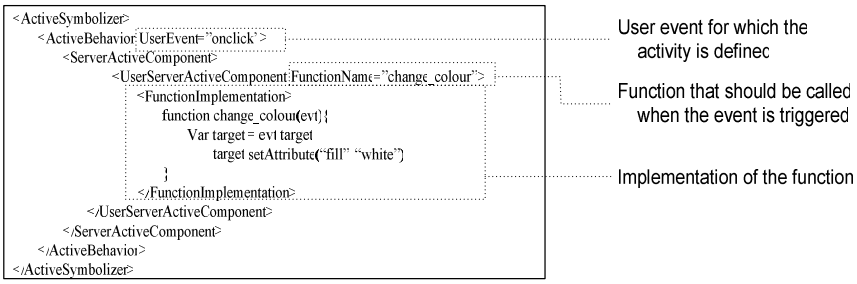



**Fig. 6.** Active Symbolizer Example

The content of a client active component is quite simple. It consists of the name of the function that has to be invoked when the event is triggered. The content of a server active component is a bit more complex. It can be of two different types: a *named server active component* or a *user server active component*. A named server active component represents a script that is known to the service in some manner. For instance, it could be predefined and available when the service starts up, or it could be added by a user and stored in a database of scripts in the service. In these cases, a user only needs to specify the name of the script. On the other hand, in a user server active component the service does not know the script that implements the behavior, which has to be provided by the user. Hence, the content of this element consists of a function name and a function implementation given as a text.

Figure 6 shows an example of the XML definition of an active symbolizer for the active map presented in Figure 4. The active symbolizer consists of a single active behavior for the user event *onclick*. This behavior is defined by a user server active component that includes a function named *change_colour* and the script that implements it.

## 5 Active WMS Description

Once the extension to the SLD language has been defined, it is necessary to describe a WMS service that uses the ASLD language for the definition of active visualization styles. We call this new type of WMS service *Active WMS* (AWMS for short). The following pseudocode represents the main processing algorithm of this type of service:

```
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 rule.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)
```

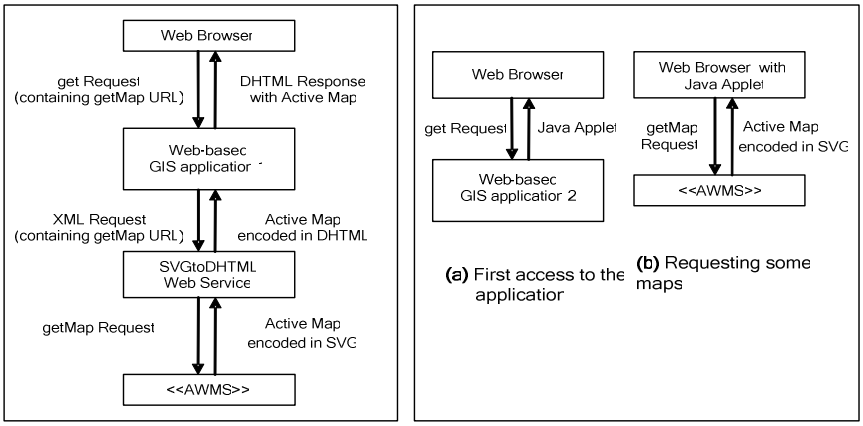
For each layer and style requested in the *getMap* request, the algorithm retrieves the active SLD definition (line 2), and selects the rules that are active for the current map scale (line 3). Then, all geographic objects from the current layer are retrieved (line 4) and each one in turn is evaluated to check whether it satisfies the filter of any rule (lines 5, 6 and 7). For each rule that the geographic object satisfies, the collection of symbolizers is retrieved and the geographic object is rendered following the visualization style defined in the symbolizer.

## 6 Client Applications Using Active Maps

The main problem related to the use of active vector formats such as SVG is that web browsers do not usually have native support for them. Therefore, it is necessary to install plug-ins that enable the visualization of those

formats in the browser. In this section, we present two examples of client applications of the AWMS service that visualize active maps inside any web browser without the need of installing any plug-in at all. The first application is based on DHTML (HTML + Javascript), whereas the second one is based on Java Applets. Since both alternatives allow the visualization of active maps inside a web browser without the need of installing any additional component, they represent an interesting advance in the field of web-based GIS applications.

Our first client application uses a web service (SVGtoDHTML) [7] that transforms any active map represented in SVG into a new active map representation that uses only DHTML elements (HTML + Javascript). This representation improves accessibility to active maps generated by an AWMS in the sense that any web browser that supports DHTML will be able to display them. This representation has also the best advantages of vector and raster formats. A raster representation of the map is used as a background image, whereas a vector representation is used to render those geographic objects that have any behavior defined. This means that the map is rendered almost as fast as a raster image while keeping the active characteristics of vector formats.




a) Using DHTML b) Using a Java Applet

**Fig. 7.** Architecture of Web Applications Using an AWMS

Figure 7(a) shows the architecture of a web-based GIS application that uses a SVGtoDHTML web service. The request to the service is encoded in an XML post request that includes the *getMap* request to an AWMS service. The SVGtoDHTML service retrieves the active map from the AWMS and returns the equivalent DHTML representation, which can be

displayed directly by any web browser. Figure 7(b) depicts the architecture of a different type of web-based GIS application. In this case the web-based GIS application uses a Java Applet to visualize active maps in the web browser of the client. The first time the application is used an applet is downloaded automatically. This applet is able to render the active map and execute its behavior in response to the user events.

## 7 Conclusions and Future Work

The main advantage of the active maps defined in this paper is that the geographic objects displayed in them can react to user-triggered events. This is achieved by associating behavior to each geographic object by means of a script function.

In order that this functionality is integrated seamlessly in web-based GIS applications we have proposed an extension to both the WMS and the SLD specifications. We defined the ASLD language as a superset of the SLD language that includes a new type of symbolizer called *active symbolizer*. This symbolizer is used to represent the association of a particular behavior, represented as a script function, with a specific user event. Script functions can be provided either by the client application or the service, and in this case, they can be functions known by the service or given by the client application in the request. It is important to remark that our proposal does not modify the specifications. Instead, we extend them in the sense that a service implementing the AWMS and ASLD specifications still conforms to the WMS and SLD specifications. It can be used in a transparent way by existing WMS client applications. We have developed an implementation of the AWMS and SLD specifications and we are working on the final tests before making it public.

We have also presented two applications that use an AWMS service to obtain active maps. The requirement of installing plug-ins in web-browsers to enable the visualization of vector graphic formats is a well-known problem . Plug-ins are the most efficient way to extend the functionality of a web browser but they present two important disadvantages: i) administrator permission are usually needed during the installation of any plug-in because of their potential security problems and ii) they are usually highly dependent on the web-browser and the technology (e.g., Internet Explorer vs. Mozilla, and Windows vs. Linux vs. Mac OS). The applications we presented use two different alternatives to overcome this problem: one uses a service that transforms SVG into DHTML and the other uses a Java Applet. Using a service that transforms SVG into DHTML has the advan-

tage that the client application is very accessible in the sense that any web browser supporting DHTML can be used. The second approach, using a Java Applet, is more efficient and gives additional functionalities due to the use of Java instead of Javascript. However, it keeps the drawback that a Java Virtual Machine has to be installed in user's computer, which still is a weaker requirement than those posed by plug-ins.

Finally, as future work, we are considering the formalization of the AWMS specification and its proposal to the Open Geospatial Organization. This would enable its use in web-based GIS applications as a source of active maps.

## References


1. Open Geospatial Consortium. Open Geospatial Consortium Specifications. Retrieved February 2007 from http://www.opengeospatial.org
2. Open Geospatial Consortium. Web Map Service Specification. Version 1.3. August 2004. Retrieved from http://www.opengeospatial.org in February 2007
3. Open Geospatial Consortium. Styled Layer Descriptor. Version 1.0.0. September 2002. Retrieved from http://www.opengeospatial.org/ in February 2007
4. World Wide Web Consortium. Scalable Vector Graphics (SVG) 1.1 Specification. January 2003. Retrieved from: http://www.w3.org/TR/SVG11/ in February 2007
5. Open Geospatial Consortium. Filter Encoding. Version 1.0.0. September 2001. Retrieved from: http://www.opengeospatial.org in February 2007
6. Thomas Brinkhoff. Towards a Declarative Portrayal and Interaction Model for GIS and LBD. In Proceedings 8th Conference on Geographic Information Science (AGILE 2005), Estoril, Portugal, 2005, pp. 449-458.
7. 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), pp. 490-494. Copenhagen (Denmark) 2005. IEEE Computer Society.
8. Nieves. R. Brisaboa, Antonio Fariña, Miguel R. Luaces, José R. Paramá, Miguel R. Penabad, Ángeles S. Places, José R Viqueira. Using Geographical Information Systems to Browse Touristic Information. IT&T: Information, Tourism and Technology, vol. 6, num. 1, pp. 31-46. USA, 2003.