

An Open GeoSpatial Standards-Enabled Google Earth Application to Support Crisis Management

Scott Pezanowski, Brian Tomaszewski, Alan M. MacEachren
The GeoVISTA Center, Department of Geography, The Pennsylvania State University, 302 Walker Building,
University Park, PA 16802 USA Ph: 814-865-3433 Fax: 814-863-7943
Email: <spezanowski, bmt139, maceachren>@psu.edu

ABSTRACT

Google Earth (GE) and related open geospatial technologies have changed both the accessibility of and audience for geospatial information dramatically. Through data rich applications with easy to use interfaces, these technologies bring personalized geospatial information directly to the non-specialist. When coupled with open geospatial data standards, such as Web Map Services (WMS), Web Features Services (WFS), and GeoRSS, the resulting web-based technologies have the potential to assimilate heterogeneous data from distributed sources rapidly enough to support time-critical activities such as crisis response. Although the ability to view and interact with data in these environments is important, this functionality alone is not sufficient for the demands of crisis response activity. For example, GE's standard version currently lacks geoanalysis capabilities such as geographic buffering and topology functions. In this paper, we present development of the "Google Earth Dashboard" (GED), a web-based interface powered by open geospatial standards and designed for supplementing and enhancing the geospatial capabilities of GE. The GED allows users to create custom maps through WMS layer addition to GE and perform traditional GIS analysis functions. Utility of the GED is presented in a use-case scenario where GIS operations implemented to work with GE are applied to support crisis management activities. The GED represents an important first step towards combining the ubiquity of GE and geospatial standards into an easy-to-use, data rich, geo-analytically powerful environment that can support crisis management activity.

Key Words: Open Geospatial Standards, Google Earth, GIS, Crisis Managemet

1. Introduction

Geographical information and technologies are essential to crisis management activities (National Research Council, 2007). Easy to use, data rich environments such as Google™ Earth (GE) are making geospatial technologies accessible and customizable to non-specialist end users for crisis management applications (Nourbakhsh, 2006). Furthermore, the proliferation of data availability through open geospatial standards such as Web Map Services (WMS), and Web Features Services (WFS) creates the potential for rapid assimilation of heterogeneous data sources to support disaster response (Open Geospatial Consortium, 2005). In this paper, we present our preliminary results on the "Google Earth Dashboard" (GE-Dashboard), a web-based interface to Google™ Earth powered by open GIS standards and the Adobe Flex open-source software environment. The GE-Dashboard has been designed to supplement and enhance the existing geospatial capabilities of Google™ Earth (GE) with functionality to incorporate WMS layers into GE and to perform traditional GIS spatial analysis functions. We discuss the technical approach used to create our solution, provide a use-case scenario of the GE-Dashboard, and offer avenues for further research.

2. Problem Domain

Although the ability to view data in GE is important, simple viewing functionality alone is not sufficient for the demands of time-pressured disaster response. For example, GE currently lacks geo-analysis capabilities such as geographic buffering and topology functions that may be used to support situation assessments during a disaster, such as to derive casualty counts in a region or track the population impacted by the spread of chemical plume. Google™ Earth (GE) demonstrated its effectiveness as a geospatial disaster response tool and platform during the Hurricane Katrina crisis of 2005 (Google Earth Blog, 2005; Google, 2007) and continues to be used for a wide variety of crisis management activities. It is an easy-to-use virtual globe environment with an eye-catching design that allows users to efficiently zoom and pan around the globe and add data layers from both Google and other groups through the Keyhole Markup Language (KML)¹. Recognizing the effectiveness and value of the free, public version of GE to support crisis management, we sought to expand the geospatial capabilities of GE without comprising the ease of use that GE affords.

3. Solution

The solution we developed to create the GE-Dashboard consists of seven high level architectural and conceptual components. These include the Open Geospatial Consortium (OGC) Web Map Services (WMS), Web Feature Services (WFS) and Stylized Layer Descriptors (SLD) standards, the Adobe Flex web development platform, the open source Geospatial database postGIS, the open source geospatial web server GeoServer, and networking components of GE itself. In this section, we provide brief sketches of each of these components to provide context for how each of these in turn were used to create the GE-Dashboard.

3.1 OGC Web Map Services (WMS), Web Feature Services (WFS) and Styled Layer Descriptors (SLD)

The Open Geospatial Consortium, Inc. (OGC)² is a non-profit, international, voluntary consensus standards organization that is leading the development of standards for geospatial and location based services. Standards maintained by the OGC allow for open and extensible software application programming interfaces to be used in geographic information applications.

The Web Map Service (WMS) standard³ allows maps of spatial data to be produced as images, typically in PNG or GIF formats. End users can view WMS maps using a simple client such as a web browser. Data that are served up as images provide the advantage that the underlying data can not be altered or stolen. However this also limits user interaction with data. The Web Feature Service (WFS) standard⁴ typically produces maps of spatial data in an XML format known as the Geographic Markup Language (GML)⁵. Because data are being returned in an XML format, a thicker client is needed to render the geography encoded in GML. An advantage with WFS is that users can query and style data at will and transactional WFS allow users to save data to a server. However, large volumes of WFS data can overwhelm limited bandwidth. Styled Layer Descriptor (SLD)⁶ encoding is an XML-based specification for implementing user defined descriptions of how geographic data is rendered in terms of colors and symbols.

3.2 The Adobe Flex web development platform

Adobe® Flex⁷ is a rich-internet application development framework. It is based on Adobe Flash®. Flex differs from Flash in that it has been designed to develop full, web-client applications and has native support for web-service consumption and database access. This differs from Flash, which is primarily used for creating animations and other vector-based graphic applications. Elegant and pleasing web applications can be quickly developed with Flex. Although maintained by a commercial software company, a free Flex SDK is available for download, thus making source code written in Flex available for open-source community use.

One drawback of using Adobe Flex is that the user needs to have the Flash Player version 9 installed. This may cause limitations such as the Adobe Flash Player not being supported on the Linux OS (GE is also not available for the Linux OS). However, the Flash player is available on 96% of all internet-capable computers, and thus makes a good choice for browser plug-in based applications (Adobe Systems, 2007).

3.3 postGIS

postGIS⁸ is an open-source software component that “spatially” enables the object-relational PostgreSQL database. For example, with postGIS, geographic objects such as lines, points, and polygons can be added to standard database tables. postGIS also provides functionality for spatially enabled SQL queries. postGIS follows the OGC’s Simple Feature Specifications for SQL standard⁹.

3.4 GeoServer

GeoServer¹⁰ is an open source web mapping server that allows geospatial data to be served over the web. GeoServer uses the WMS and WFS protocols for serving data in image and XML formats from a variety of sources that include shapefiles and postGIS-enabled databases.

3.5 The Google Earth Network Link Object

The Google Earth Network Link object11 allows KML data to be accessed in GE from an outside URL or other remote location. It is a critical component to custom GE applications that need to add external data in real-time.

4. GE-Dashboard

4.1 Overview

The GE-Dashboard is a web-based interface, developed in Flex and designed to accompany GE. Since it is a web-based interface, the GE-Dashboard is loaded into the built-in web browser control of GE to create a seamless interface. The GE-Dashboard allows users to add their own custom WMS Layers and allows users to perform GIS functionality on those layers in GE. The overall architecture of the GE-Dashboard can be seen in Figure 1.

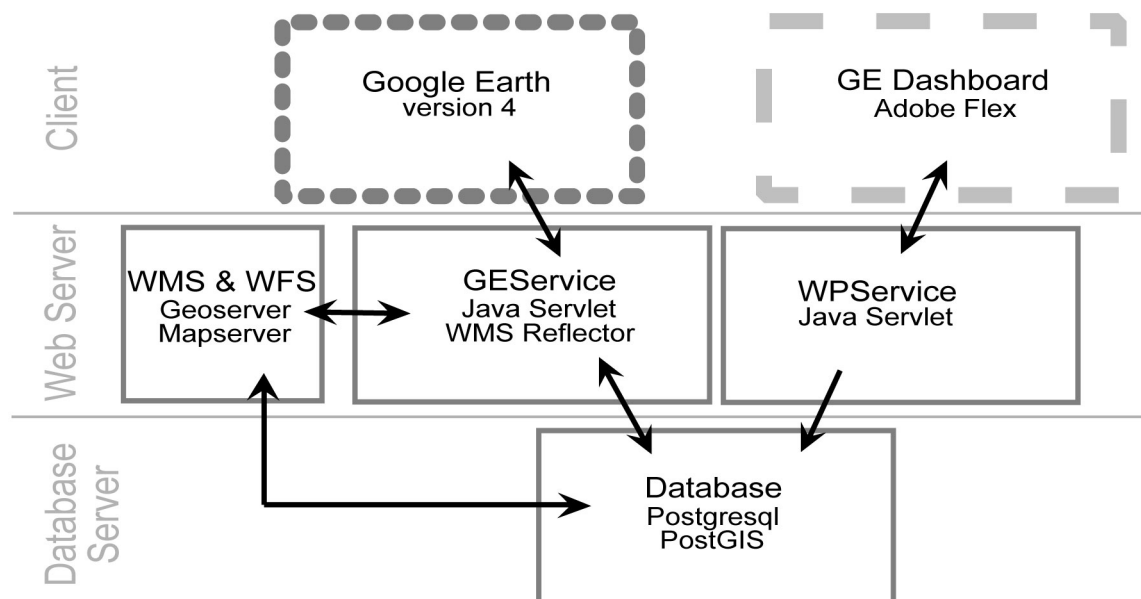
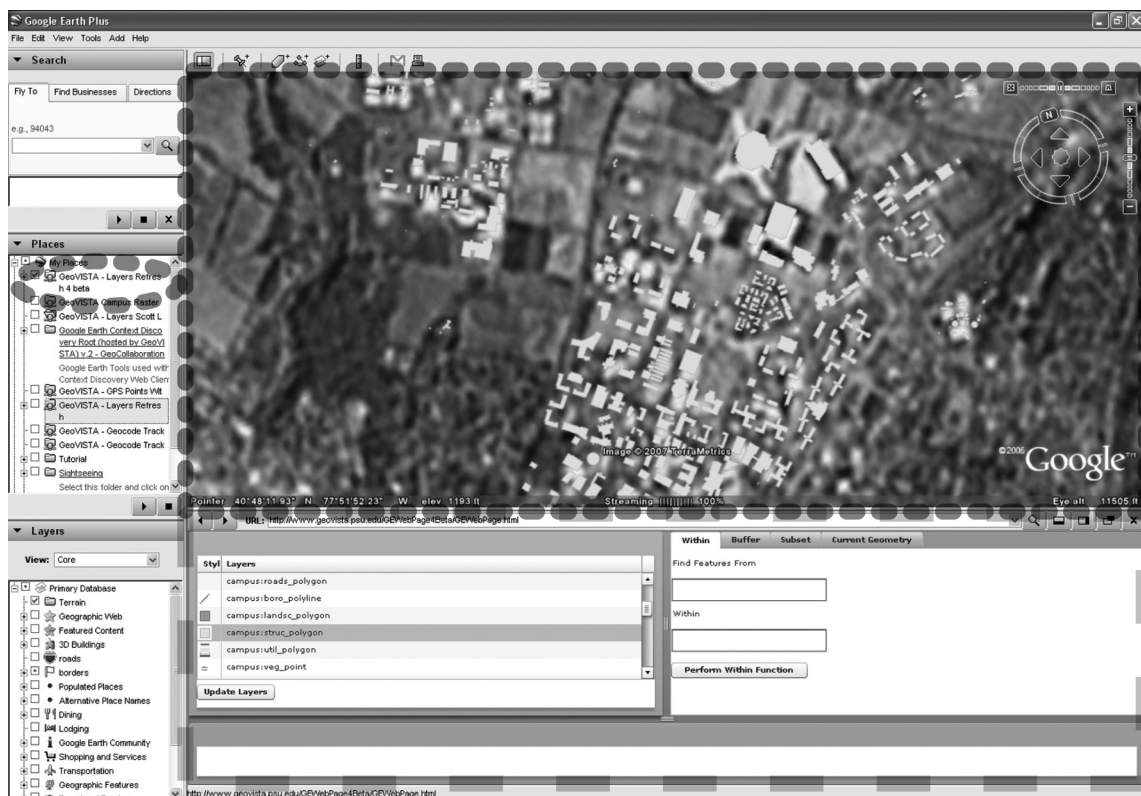


Figure 1 – The overall architecture of the GE-Dashboard. Dotted boxes in bottom schematic diagram represent actual interface components outlined in the application screenshot above.

In this figure, two important components are shown that allow the GE-Dashboard to function. First is the custom WMS Reflector for GE. Second is a database which serves as both a back-end and middle-ware component that GE and the GE-Dashboard communicate with.

In its basic form, a WMS reflector is a server application that is accessed by a client such as GE, and communicates with a WMS-enabled server like Geoserver. In our implementation a request from GE comes through the network link in to the WMS Reflector. The WMS Reflector then requests a WMS map image from Geoserver. Geoserver then returns this image to GE where the image is overlaid onto the main GE globe.

The GE-Dashboard creates an interface that makes it easy to add WMS layers, turn WMS layers on and off and also allow users to visually see results of a GIS function on the GE globe through the dynamic styling of the WMS layer (such as highlighting features returned by a spatial query).

The back-end database of the GE-Dashboard records user action. For example, if the user decides to view two particular WMS layers, the layer names and their SLD styles are recorded in the database along with some ancillary data such as the IP address of the users' computer. This function ensures that the correct layers are drawn for the correct user, as GE can not distinguish between different users accessing the back-end database.

4.2 GIS functionality

A key aspect of the GE-Dashboard is the users' ability to perform GIS analysis functionality with the data present. One function supported currently in the GE-Dashboard is a within function. A drag-drop graphical interface was implemented in Flex to perform this function. A detailed description of how a user of the GE-Dashboard can use the GIS functionality is provided below (in section titled "GE-Dashboard Process Flow").

GIS functionality of the GE-Dashboard is provided by sending custom WFS requests to Geoserver, which are then passed on to the backend postGIS database for processing, thus load-balancing all of the computationally intensive work onto the server. This allows the results of the GIS function to be returned quickly to the client.

Geoserver provides an eloquent way to access the back-end database through a web client, and serves as a middle-ware between the client GIS functionality request and back-end processing of those requests. In particular, a WFS within function request is similar to that shown in Sample Code 1, below.

```
<?xml version="1.0" encoding="utf-8"?>
<wfs:GetFeature service="WFS" version="1.0.0"
  outputFormat="GML2"
  xmlns:campus="http://www.geovista.psu.edu/campus"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs
http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd">
<wfs:Query typeName="campus:structures">
<ogc:Filter>
<ogc:Within>
  <ogc:PropertyName>the_geom</ogc:PropertyName>
  <gml:MultiPolygon... {rest of Geometry here}
  </ogc:Within>
</ogc:Filter>
</wfs:Query>
</wfs:GetFeature>
```

Sample Code 1 - Sample WFS Query Request.

In this case, the campus:structures layer is queried for all of the features within the Geographic Markup Language (GML) geometry. This xml is posted to Geoserver's WFS, where the request is translated into SQL to

query the postGIS database. Geoserver also translates the records returned from the query back into GML that is then returned to the client.

In the WFS query process, the GE-Dashboard constructs the query from a GML template and adds user input. Adobe Flex makes it programmatically easy to post this request to Geoserver and parse the results through Flex's native HTTPService Object. An HTTP post in Adobe Flex looks much like Sample Code 2.

```
<mx:HTTPService id="filterFunctionRequest" showBusyCursor="true"
url="http://www.geovista.psu.edu/geoserver/wfs" result="showResults(event)" method="POST"
contentType="application/xml" resultFormat="e4x" useProxy="false" >
    <mx:Request>
        WFS GML here.
    </mx:Request>
</mx:HTTPService>
```

Sample Code 2 - Sample Adobe Flex HttpService tag POST request.

4.3 SLD Service

Another important aspect of the GE-Dashboard is the SLD Service. This service provides dynamic cartographic styles and symbols to the WMS layers appearing in GE. The purpose of the SLD Service is to accept SLD xml which is posted to this service from a client application, and to then write this SLD xml to a file on the server which can be accessed from the web and Geoserver. Finally, it returns the url of this SLD file to the client application.

With this approach, the GE-Dashboard can pass Geoserver the SLD file url and dynamically style any WMS layer. The url for dynamically styling layers in Geoserver looks something like that of Sample Code 3.

<http://www.geovista.psu.edu/geoserver/wms?request=GetMap&srs=EPSG:4326&width=400&height=400&SLD=http://www.geovista.psu.edu/GEService/SLDFiles/highlightStructures.sld>

highlightStructures.sld

```
<StyledLayerDescriptor version="1.0.0">
    <UserLayer>
        <Name>campusstructures</Name>
        <UserStyle>
            <FeatureTypeStyle>
                <Rule>
                    <Name>Rule 1</Name>
                    <PolygonSymbolizer>
                        <Fill>
                            <CssParameter name="fill">#FFFF00</CssParameter>
                            <CssParameter name="fill-opacity">0.5</CssParameter>
                        </Fill>
                        <Stroke>
                            <CssParameter name="stroke">#FFFF00</CssParameter>
                            <CssParameter name="stroke-width">1</CssParameter>
                        </Stroke>
                    </PolygonSymbolizer>
                </Rule>
            </FeatureTypeStyle>
        </UserStyle>
    </UserLayer>
</StyledLayerDescriptor>
```

Sample Code 3 – Sample GetMap request with SLD file XML.

Geoserver does have the ability to accept the actual SLD xml as a url parameter. This method was not used because Microsoft Internet Explorer truncates urls at a specific character length. This often causes problems

when adding a lengthy xml body into the url. Therefore, the SLD Service was necessary. The SLD Service allows for a simple url to the SLD file to replace the actual SLD body xml.

4.4 GE Network Link

The GE-Dashboard uses a GE Network Link to point to the URL of the GEService, or WMS Reflector, which is a Java servlet.

The GE Network Link requests KML from the GEService and attaches the geographic bounding box derived from the current map view in GE to this request. The GEService accepts this request and queries the database for the latest layers and styles selected by the user in the GE-Dashboard.

Subsequent to this, when a GE network link makes a request, the following actions occur in the GEService's custom WMS reflector. When a request comes into the GEService from the GE Network Link, the first action taken by the GEService is to find out if the latest record layers/styles requested from the users of GE-Dashboard have been handled (if the user of the GE-Dashboard has changed layers, performed a GIS function, or any other action that requires a new map). If the database record has not been handled, then the GEService sets the HTTP expiration information to the date/time the user changed layers (in a sense, in the past) (Sample Code 4). The entire GE-Dashboard process is described in the section titled "GE-Dashboard Process Flow."

```
response.setDateHeader("Expires", date.getTime());
```

Sample Code 4 – Sample Java servlet code for setting the HTTP Date header to expire.

By setting the HTTP Header information to a date in the past, the GEService tells the GE Network Link to refresh the KML the network link is storing, and subsequently gets the latest map based upon user interaction in the GE-Dashboard. Conversely, if the latest record of user interaction in the database has already been handled, then the GEService sets the expiration as negative one (see Sample Code 5).

```
response.setDateHeader("Expires", -1);
```

Sample Code 5 – Sample Java servlet code for setting the HTTP Date header to never expire.

This effectively tells the GE Network Link that the HTTP header information in the servlet has not expired and the network link does not need to refresh.

The interactions between the GEService and the GENetwork link take advantage of a new feature first available with GE version 4 and KML 2.1, the `<refreshMode>onExpire</refreshMode>` tag. This tag makes several GE-Dashboard functions practical. Specifically, this KML tag tells the network link to refresh itself when the HTML header information of the page it points to expires, thus allowing constant data updates.

4.5 GE-Dashboard Process Flow

The process flow that allows the GE-Dashboard to function includes eleven steps. (1) The user changes the layer(s) in the GE-Dashboard layer list (or performs a GIS function causing a new layer or style to be displayed). (2) The user clicks the "Update Layers" button on the GE-Dashboard. (3) The GE-Dashboard sends a request to the WPSERVICE with the layers and styles to be displayed. (4) The WPSERVICE writes this information to the database along with the IP address of the client computer, the time stamp of the request and a Boolean false variable that the record has not been handled by GE.

(5) Shortly later, the GE Network Link pings the GESERVICE to see if the page has expired. (6) The GESERVICE queries the database to see if the latest record for the client machine has been handled. (7) Since the user has made a change in layers, the GESERVICE tells the GE Network Link to continue with its request for the latest layers and styles. (8) The GESERVICE changes the Boolean of the record to list that the record has been handled. (9) The GESERVICE knows the layers and styles requested by the client from the previous query. (10) It can then construct the KML to be returned to the GE Network Link. (11) Finally, KML is returned to the GE Network Link which tells GE to draw the image overlay based upon the request layers and styles.

Although, this process is complicated "behind-the-scenes", the UI is quite easy to use. The user chooses layers to be displayed, clicks the "Refresh Layers" button and the layers are displayed on the GE globe. Or, the user performs a GIS function and the features that are resulting from the function are highlighted on the GE globe.

5. Use Case Scenario

This section outlines a prototypical use case for the GE-Dashboard. The use case, presented in scenario form below, focuses on a toxic release crisis in which output from a plume model is integrated with other information in order to make response decisions.

Scenario: There is a chemical leak on a University Campus. This chemical leak is airborne and could potentially affect thousands of students, faculty and staff. The disaster manager brings up GE and adds the necessary layers to the visualization to get an overview of the situation. The manager next drags the building footprint layer into the within functions' first text box. Then the plume model polygon layer depicting the chemical leak is dragged into the second text box of the within function. This allows the user to find all building footprints or structures within the plume model. The buildings identified are highlighted on GE and the attributes of the buildings are shown in the GE-Dashboard. The building footprints layer is joined to a table that has the Registrar's estimate for population in each building at the current time. This allows the manager to develop a plan for evacuation of potentially affected buildings.

6. Future Research

We foresee several directions that developments with the GE-Dashboard can take.

The GE-Dashboard is a web-based interface designed to be used with GE's built in web browser. This allows the GE-Dashboard to work effectively as a seamless application with GE. However, because the GE-Dashboard is a web application, it could easily be opened in an external browser and its functionality made available to any other client mapping application that can display OGC-standards compliant data. Furthermore, the GE-Dashboard could also be completely embedded as a component in any other Adobe Flex-based client that needs mapping support. We are currently experimenting with these variants on the general model.

Future research also involves extending the GE-Dashboard to include additional GIS functionality. Both postGIS and Geoserver support many other GIS Functions that could be connected to the GE-Dashboard (e.g, buffering, intersection). One challenge of implementing additional GIS functionality is the design of easy to use, intuitive interfaces to access different types of GIS functions through the GE-Dashboard.

In addition to extensions to functionality of the system, there is potential to use the GE-Dashboard to study use of web mapping tools in crisis management activities. The current GE-Dashboard stores records of user activity in a database. This record can support study of how users interact with the interface when carrying out crisis-related tasks (Robinson & Weaver, 2006). This could help improve the interface as well as aid in training during crisis management exercises.

7. Conclusion

The GE-Dashboard represents an important first step towards combining the ubiquity of GE with open geospatial standards into an easy-to-use, data rich, geo-analytically powerful environment that can support disaster and risk management activity. Further research developments using this approach can lead to improved crisis management systems that can overcome challenges in information availability and interoperability across heterogeneous software environments, development platforms, and sources of data.

Acknowledgements

The research reported here has been supported by the National Science Foundation under Grant EIA-0306845. This work is also supported by the National Visualization and Analytics Center, a U.S. Department of Homeland Security program operated by the Pacific Northwest National Laboratory (PNNL). PNNL is a U.S. Department of Energy Office of Science laboratory.

References

Adobe Systems (2007), Flash Player Statistics. Available: http://www.adobe.com/products/player_census/flashplayer/2 March 2007.

Fuhrmann S. & Pike W (2004) User-centered Design of Collaborative Geovisualization Tools. In J. Dykes & A. MacEachren & M. J. Kraak (Eds.). Exploring Geovisualization (pp. 591-609): Elsevier.

Google (2007) Hurricane Katrina Imagery. Available: <http://earth.google.com/katrina.html2> March 2007.

Google Earth Blog (2005) Hurricane Katrina Archives. Available: http://www.earthblog.com/blog/archives/hurricane_katrina/2 March 2007.

National Research Council (2007) Successful Response Starts With a Map: Improving Geospatial Support for Disaster Management. Washington, D.C.: National Academies Press.

Nourbakhsh I. 2006, Mapping disaster zones. Nature, pp. 439.

Open Geospatial Consortium (2005) OGC announces Risk and Crisis Management Working Group. Available: <http://www.opengeospatial.org/pressroom/pressreleases/4223> March 2007.

Robinson A C & Weaver C (2006) Re-Visualization: Interactive Visualization of the Process of Visual Analysis. Workshop on Visualization, Analytics & Spatial Decision Support at the GIScience conference, Münster, Germany.

Tomaszewski B (2003) Emergency Response and Planning Application Performs Plume Modeling. ArcUser, 6, pp. 10-12.

1 KML is Google Earth's proprietary markup language used for adding symbols to the GE map, and performing other functions

2 <http://www.opengeospatial.org/>

3 <http://www.opengeospatial.org/standards/wms>

4 <http://www.opengeospatial.org/standards/wfs>

5 <http://www.opengis.net/gml/>

6 <http://www.opengeospatial.org/standards/sld>

7 <http://www.adobe.com/products/flex/>

8 <http://postgis.refrations.net/>

9 <http://www.opengeospatial.org/standards/sfs>

10 <http://docs.codehaus.org/display/GEOS/Home>

11 http://earth.google.com/kml/kml_tags_21.html#networklink