Modular Web-Based Atlas Information Systems

Bernhard Jenny
Institute of Cartography / ETH Zurich / Zurich / Switzerland

Andrea Terribilini
OneOverZero / Zurich / Switzerland

Helen Jenny
Institute of Cartography / ETH Zurich / Zurich / Switzerland

Radu Gogu
Department of Geotechnical Engineering and Geo-sciences / School of Civil Engineering / Technical University of Catalonia / Barcelona / Spain

Lorenz Hurni
Institute of Cartography / ETH Zurich / Zurich / Switzerland

Volker Dietrich
Institute for Mineralogy and Petrography / ETH Zurich / Zurich / Switzerland

Abstract

Atlas information systems (AIS) present spatial information on predefined themes and localities in the form of maps and other representations, generally focusing on correct cartographic appearance and offering a certain degree of user interaction. This article introduces the concept of modular AIS, a concept that is essential for the development of an AIS with modern computer technology. The main advantages of a modular architecture are twofold: first, an AIS software framework based on a modular architecture allows for easy and rapid customization to a certain theme and locality; second, functional enhancements and new technologies can be easily integrated into a modular AIS in order to optimally present and analyse the data at hand.

Web-based AIS can benefit from the concept of modularity at three different levels: (1) The AIS client can adjust its functionality and adapt to the available technology on the present computer platform. (2) The AIS server can build an AIS client with custom-tailored data and functionality in real time, depending on the user's access rights, needs, or expertise. (3) Distributed, modular data storage greatly simplifies the design, implementation and maintainance of an AIS by using a mediation system.

To illustrate the concepts presented, we will discuss selected technical aspects (e.g., Web-based map viewer technology, client–server communication), and describe an exemplary Web-based AIS that extends the modular core architecture through specialized functionalities for the analysis of geophysical data. It is the authors' hope that the ideas presented will provide an introduction to the technical concepts for designers and developers of similar Web-based atlas information systems.

Keywords: interactive web cartography, customizable, distributed, component-based mapping architecture, Java, SVG, natural hazards

Résumé

Les systèmes d'information cartographique (AIS) schématisent des données spatiales sur des localités et des thèmes prédéfinis sous forme de cartes ou d'autres modes de représentation, orientés en général vers une apparence cartographique précise et attribuant un certain degré d'interaction à l'utilisateur. Le présent article présente le concept d'un AIS modulaire, en tant que concept essentiel au développement d'un AIS lié à la technologie informatique moderne. Deux des principaux avantages d'une architecture modulaire sont d'abord un cadre logiciel AIS basé sur une architecture modulaire qui permet de personnaliser facilement et rapidement un certain thème et une certaine localité, et ensuite l'intégration facile des améliorations fonctionnelles et des nouvelles technologies dans un AIS modulaire afin de d'analyser et de présenter les données existantes de façon optimale.

Un AIS sur le Web peut tirer avantage du concept de modularité à trois niveaux : 1) le client AIS peut régler sa fonctionnalité et s'adapter à la technologie disponible sur la plateforme informatique actuelle; 2) le serveur AIS peut créer un client AIS avec des données sur mesure et une fonctionnalité en temps réel en fonction des droits d'accès, des besoins, ou de l'expertise de l'utilisateur; et 3) le stockage modulaire des données réparties simplifie la conception, la mise en oeuvre et la maintenance d'un AIS à l'aide d'un système de médiation.

Pour illustrer les concepts présentés, nous discutons d'aspects techniques sélectionnés (par ex., technologie de visualisation cartographique sur le Web, communication client-serveur), et donnons un exemple d'un AIS sur le Web qui étend l'architecture modulaire de base par le truchement de fonctionnalités spécialisées pour l'analyse des données géographiques. Les auteurs espèrent que les idées présentées serviront d'introduction aux concepts techniques pour les concepteurs et les développeurs de systèmes d'information cartographique similaires sur le Web.

Mots clés : cartographie web interactive, architecture cartographique adaptable et distribuée basée sur des composants, Java, SVG, dangers naturels

Today's Atlas Information Systems

Enhancing computer-based atlas information systems (AIS) with the concept of modularity offers many advantages, the two most important being effortless adaptation of the AIS to new themes and geographic areas and easy extension of an AIS with specialized functionality. This article will present the proposed modularity concept that was developed for Web-based AIS before describing the details of the proposed enhancements.

The term ''geographic information systems'' (GIS) does not refer to a single, clearly defined type of software application. Instead, it encompasses a family of different applications that share the common aspects of storing, mapping, and analysing spatial data. An atlas information system (AIS) can be considered a type of GIS that differs substantially in certain respects from conventional GIS: ''Atlas Information Systems are computerized geographic information systems related to a certain area or theme in conjunction with a given purpose – with an additional narrative faculty, in which maps play a dominant role'' (van Elzakker 1993, cited in Ormeling 1995, 2127). Analogous to the classical paper atlas, an AIS presents spatial information on predefined themes and localities. Using digital media, the AIS provides information in the form of maps, alternative spatial representations, and multimedia content. The user influences map content and appearance by combining map layers and changing visualization parameters. The system may also allow the user to integrate personal geo-information. An AIS must offer an interface that inexperienced users can easily grasp. To promote user interest, and so as not to overstress the user's patience, system response time should be fast, a requirement that disqualifies slow-performing simulations and computations. Barbara Schneider (1999) provides a more detailed characterization of atlas information systems.

The distribution of AIS relies on two methods: (1) physical media, such as CD-ROM; or (2) Internet retrieval. We concentrate in this article on AIS provided via the Internet, which we classify as follows:

- *Entirely Web-based AIS* are client–server applications that rely on the Internet to transmit all data and program code. The client component is integrated into a Web page and uses a Web browser as the host application. For the system programmer, this integration reduces the range of viable technologies and programming languages, since only those supported by Web browsers are available. Entirely Web-based AIS do not require users to install software applications on their computers. Instead, the Web browser always receives the most up-to-date data and system functionalities. Despite these advantages, however, the limited bandwidth of the Internet still hinders the distribution of large data sets, including those involved in distributing AIS.
- *AIS with online update capability* require initial installation from a CD-ROM or similar medium (Hurni, Bär, and Sieber 1999). Up-to-date information is subsequently downloaded from a server, allowing the system author to distribute updated or corrected maps. Since such an AIS may run independently from a Web browser, the programmer can

choose from a wider range of different technologies and programming languages.

Modularization of AIS

In the previous section of this paper we reiterate that today's Web-based AIS offer a user-friendly graphical interface and emphasize the quality of cartographic presentation. Their functionality and data are (fully or partially) distributed throughout the Internet.

Modularity is an advantageous enhancement of current AIS. A modular software architecture consists of a core framework combined with a set of task-specific modules. The author of a new AIS can select the required modules and combine them with the core framework to build a custom-tailored AIS. The resulting system offers exactly the functionality that the user needs. If an AIS requires a specialized functionality that is currently not offered by any module, an additional module must be developed. The new module must adhere to well-defined communication rules and behavioural patterns. This ensures that the core framework can handle any module, independent of its specific functionality.

An Internet-based AIS usually follows the classical client–server paradigm. Modularity can be successfully applied to the implementation of both parties. On the server side, modular software architectures such as Enterprise Java Beans (EJB) are widely used today, not only by GIS-related server applications but also by almost every imaginable area of computing, because they greatly facilitate the development of new software.

In contrast to the well-established server-side modularity, AIS clients generally do not use modular architectures. Modularity allows for the customization of client software by integrating only the required functionality into the final client. This offers three main advantages:

1. The size of the client software is minimized, which reduces download time.
2. The user interface contains only indispensable elements, which makes the AIS easy to learn and control. Our example client-side framework uses two types of modules: (1) general-purpose mapping modules that accomplish functions required by most AIS (e.g., map layer composition or navigation in a map) and (2) specialized modules that extend the functionality of an AIS in a particular area of expertise.
3. Modularity also allows for the construction of customized applications in real time. A server can add selected modules to the framework depending on the user's access rights, needs, or expertise. The user then receives a personalized AIS "on demand."

Data storage is another important area that can be extended by the concept of modularity. Distributed (or modular) data storage can facilitate maintenance of AIS data. Modular data-storage systems are generally accessed through a central access point provided by a server that hides the underlying modularity (or distribution) of data sources.

In the context of this article, therefore, we define a modular Web-based AIS as follows: A modular Web-based AIS is an AIS as defined by van Elzakker (1993, cited in Ormeling 1995) that additionally possesses a customizable, modular system architecture and the capability of retrieving data and system functionalities via the Internet.

A Modular Web-Based AIS Client

The modular Web-based AIS client presented in this section can be easily and rapidly adapted to different types of applications. Its core framework implements the basic elements of an *entirely Web-based* AIS. Our AIS client uses vector data for map rendering and accesses distributed sources to retrieve data. A series of modules extends the framework and offers the user AIS functionality combined with a graphical user interface.

AIS for the Internet

Our Web-based AIS follows the client–server architecture with a series of clients connecting to one central server. The client can download raw attribute data and transform them to interactive and animated maps, diagrams, and other presentations. The client does not necessarily have to perform all the computation that an analysis or visualization requires. Time- or data-intensive tasks can also be delegated to an application server in order to accelerate computation and reduce data transfer time.

In the developed system, the client is integrated into a platform-independent Internet browser. The core framework uses Java technology. Java is a cross-platform, object-oriented programming language, complemented by a powerful set of functionalities. To the developer of an AIS, it offers important functionalities such as networking, security management, object serialization, database connection, and data retrieval. Most importantly, it facilitates the design of modular software architectures.

Customizing an AIS Client by Modules

Our client consists of a core framework and a set of standard and specialized modules. This architecture allows for flexible adaptation of the system to different AIS scenarios (i.e., customized AISs for different users)

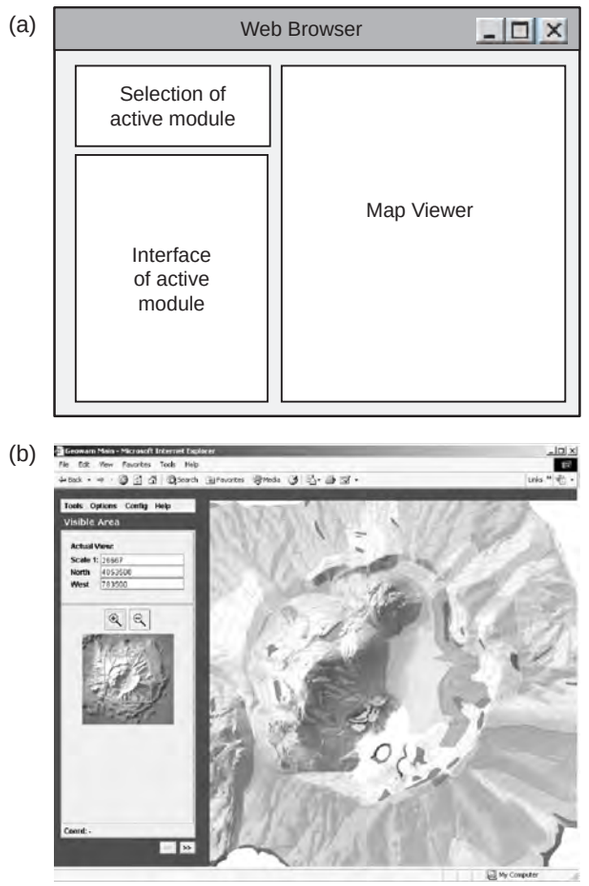

Figure 1. The active module and the map share space on the Web browser interface. Above: Schema of the active module and the Map Viewer. Below: Screenshot of the GEOWARN AIS (described in the last section of the article).

and different localities or themes. When loading the access page, the Web server assembles the needed modules in order to form the custom-tailored application.

After the modules are assembled with the core framework, the client software is delivered to the client's computer. The framework is then responsible for activating exactly one module at a time. The active module provides the currently visible graphical user interface and interacts with the map. The user of the AIS can choose the active module using a graphical control element (Figure 1). Upon user request, the framework deactivates the currently active module and configures and activates the newly chosen one (Figure 2). This mechanism requires the modules to comply with a straightforward application programming interface (API). The API is used to load, activate, and configure modules.

The framework offers a pool of services to the currently active module (Figure 2). For instance, the Resource

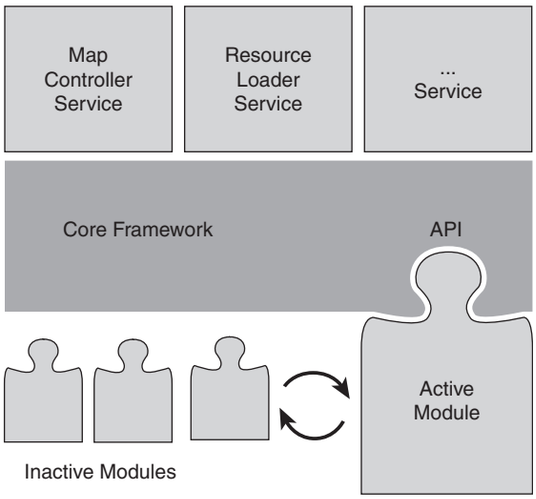

Figure 2. Framework for managing modules and services.

Loader service loads data from a server, while the Map Controller service interacts with the map; both will be discussed later. These services facilitate the implementation of additional modules, since they assume tasks that are recurrent or are critical to performance or security. Using the services reduces the complexity of additional modules and minimizes the programming effort necessary to implement them. A welcome side effect is the reduction of the application size, which, in turn, reduces download time.

MAP VIEWER MODULARITY

High-quality rendering of interactive maps in vector format requires time-consuming computations. At the time of implementation, four technologies were available for this task: Macromedia Flash (Adobe 2006), Scalable Vector Graphics (SVG; W3C 2006), Java 2D (SDN 1994–2006), and Virtual Reality Modeling Language (VRML; Web3D Consortium n.d.). Each comes with its respective advantages and problems, depending on the platform and Web browser on which it is running. We compared the four technologies during an evaluation phase (Jenny, Freimark, and Terribilini 2002). The result was that the development of a Java 2D–based viewer would have been more time intensive because of the required programming effort. Additionally, Java 2D rendering with anti-aliasing was found to be slower than rendering with Flash or SVG viewers; response time is a crucial criterion for maps that react on user actions (e.g., mouseover events that change the colour of map elements). The two-dimensional output of current VRML renderers (which are primarily optimized for three-dimensional models) could not meet our demands for high-quality graphics. Finally, SVG was favoured over

Flash because SVG is an XML-based format that is easily generated and can be edited with standard text editors. An SVG graphic may contain text, images, and Bézier line art. The Adobe SVG Viewer is the most widespread plug-in for Web browsers. It renders high-quality graphics with anti-aliasing and transparency.

Since we cannot foresee the development of future graphics standards for the Internet, the rendering module should be easily replaceable. The described framework therefore wraps the rendering engine in an exchangeable code module called Map Viewer. The Map Controller is a complementary service module that supervises the Map Viewer. As described in the previous section, the Map Controller is an example of a service provided by the core framework (see Figure 2). It manages the two-way communication between the active module and the Map Viewer (Figure 3). The active module can send commands to the Map Viewer to hide, show, add, remove, or graphically change a map element. In the other direction, the Map Viewer alerts the active module when mouse events occur (mouse-over-element, click-on-element, etc.).

The Map Controller offers a viewer-independent programming interface for communication between the active module and the Map Viewer. It relies on an exchangeable Communication Bridge that converts the Map Controller's messages to commands understandable by the Map Viewer, and vice versa. This architecture ensures effortless integration of future rendering technologies. It can also be used to select a specific viewer from a range of supported alternatives, depending on their availability on the client's computer. In the event of integrating a new type of Map Viewer, only a new Communication Bridge has to be created.

Figure 4 illustrates how the Map Controller communicates with the Adobe SVG Viewer using a specialized Communication Bridge component. Messages between the Map Controller and the Map Viewer traverse two layers that transform Java commands to JavaScript code that can be understood by the SVG Viewer (ECMAScript is the standardized version of JavaScript; see ECMA International 1999). This modular architecture can be easily adapted to other JavaScript-based viewer technologies by exchanging the second layer, labelled "JavaScript to SVG Converter" in Figure 4. For future alternative map viewers that do not understand JavaScript (e.g., a map viewer based on Java 2D), the Communication Bridge component would have to be replaced.

CLIENT–SERVER COMMUNICATION

AIS clients access their server for data retrieval. Server access can be triggered by a user action or autonomously, by a software module, when it needs additional data to analyse or display. This section explains how clients query data from a server. Access of the server to distributed data sources will be discussed in a later section.

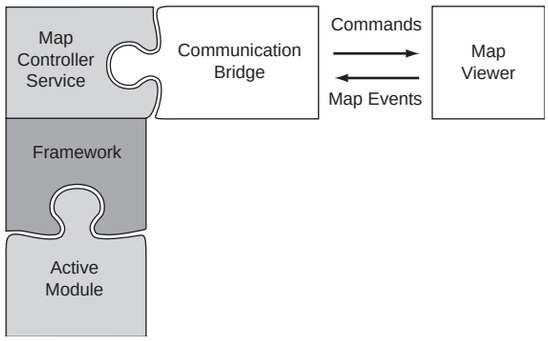

Figure 3. Communication between the active module and the map viewer.

Client–server communication is illustrated by means of an exemplary AIS client module that queries the temperature of a water spring and displays the temperature in its user interface (see upper part of Figure 5). In our example, the data flow is initialized when the user clicks on a map symbol representing a spring. The click generates an event that is transferred to the module, which subsequently requests the temperature of the spring from the server. The server returns this information, and the module displays the temperature in a text field.

Figure 5 illustrates the data flow through the different components. The Map Viewer informs the Map Controller of the event by sending it a click event (1). The Map Controller delegates this event to the currently active module (2). After analysing the event, the active module delegates the loading of the temperature data to the Resource Loader Service provided by the core framework (3). The Resource Loader Service sends a request to the server using standard HTML technique (4). The server returns the requested data in the form of an XML-formatted table (5). The Resource Loader Service passes the data to the active module that started the request (6). It is then the module's task to parse and interpret the data and present them to the user in graphical form. In our example, the resulting table contains a single value, the spring's temperature. The module finally extracts the temperature from the table and displays it in the user interface (7).

The complete process, starting with a click by the user on a spring symbol and ending with a temperature being displayed, may appear rather complicated. Yet the programmer of the module can use the services provided by the core framework. Most modules will use the Map Controller Service to interact with the map and the Resource Loader Service to communicate with the server. If needed, additional services can be used, for example,

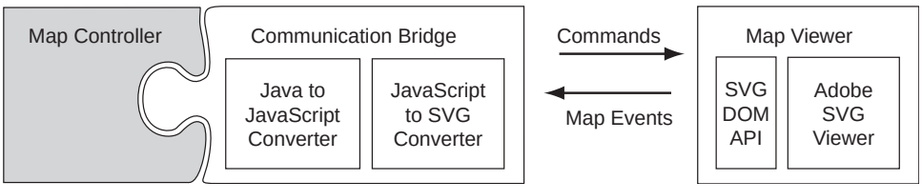

Figure 4. The Communication Bridge for the Adobe SVG Viewer.

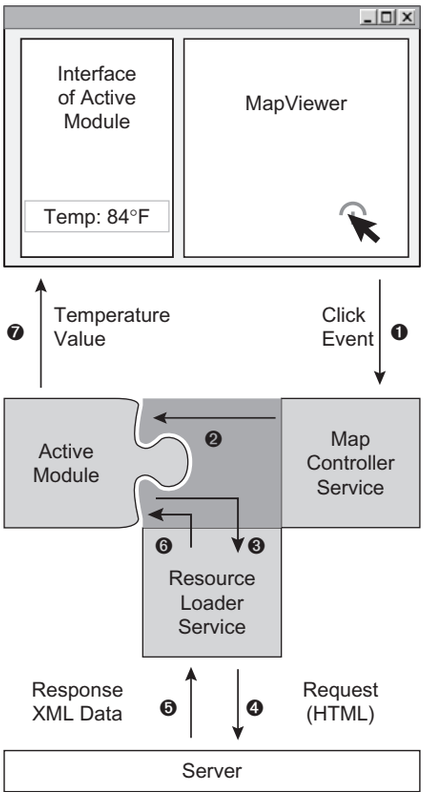

Figure 5. Data flow of an exemplary module displaying the temperature of springs.

to display data in diagrams, to analyse special types of data, or to parse XML data. The services will greatly simplify the task of the programmer, who only has to take care of the following three steps:

1. Receive mouse-click events and communicate with the Resource Loader Service to retrieve data.
2. Display the returned data in a graphical user interface.
3. Write an XML description of the query and the connection for server-side data retrieval (described in the next section).

MODULAR AIS SERVER AND DATABASE

The server side of Web-based AIS uses modularity in two distinctive areas. First, the software application running on the server and providing clients with data and services is usually based on a modular architecture. Second, data are modularly distributed to multiple databases for data storage.

The server-side software of an AIS server fulfils three AIS-specific tasks:

1. It generates the AIS clients by adding modules to the clients' core framework.
2. It provides and coordinates access to distributed data sources.
3. It supplies specialized services to client modules that require server-based functionality (e.g., data analysis or computation of visualizations that require large data sets or large computation capacity). The required functions are very specific to the AIS and its data. Furthermore, this type of modular architecture is well established and widely used (e.g., Servlets or EJB). For these two reasons, this type of server task is not discussed in this article.

ASSEMBLING AIS CLIENTS

An entirely Web-based AIS does not require users to install any software or data on their computers. Instead, the user accesses the AIS through a standard Web browser and loads a Web page that embeds the client software. The server can deliver the same assembled software to every client computer or deliver different software depending on the user's access rights, needs, or expertise. The client software can be further individualized by assembling the application at runtime. This allows for a completely customized AIS client whereby users can choose not only which data they would like to see and analyse but also which functionality they would like to use.

When assembling an AIS client, the server creates a copy (or "instance") of a module before adding it to the framework. Thus, the server can add multiple instances of a module to the framework and configure them differently. In our previous example, an AIS could use a first instance to display the springs' temperatures and a second one to display their discharges. The server would pass different data sources to the two instances. This mechanism not only allows for configuring the data source but can also be used to alter the instances'

graphical user interface. In the example, the server would pass appropriate parameters to change the graphical interface from "Temp.: 84 F" to "Discharge: 0.8m^3 per min." A module could offer even more advanced options to configure its behaviour, data sources, graphical user interface, and so on.

DISTRIBUTED DATA SOURCES

An entirely Web-based AIS usually cannot hold all potentially necessary data on the client's computer. This would strain the storage capabilities of the host browser and cause unacceptably long download periods. To overcome these shortcomings, a Web-based AIS relies on a server for storage, extraction, analysis, and synthesis of data. AIS need very different types of data that are best stored in their native formats. It may also be advantageous to store data on different servers (e.g., in spatial data infrastructures) for distributed updating and maintenance of the data. This results in programmers having to confront a wide variety of different formats, query languages, and data locations. To facilitate the development of AIS modules, a single access point is required that hides the details of the underlying data sources, structures, locations, and access mechanisms. Mediator-based data access provides this type of service.

DATA MEDIATOR

A mediator-based system includes an application layer, a mediation layer, and a foundation layer, portrayed from left to right in Figure 6 (Wiederhold 1992). In the case of an AIS, the application layer corresponds to the AIS client. It requests and receives data from the mediation layer, which is a specialized part of the AIS server. The foundation layer may consist of heterogeneous data sources that can be stored on diverse servers and use different database engines and formats. Specialized software wraps each data source and offers a homogenized view.

With this architecture in place, the AIS client can request data by sending a query to the server-side mediator system. The mediator subsequently asks for data from the foundation layer and returns the result in an easily readable XML format. Note that a single query may request data from more than one source. In such a case, the mediator dispatches the request to the data sources concerned, assembles the returned fragments, and sends this final result back to the client.

The AIS described here stores the details of each request on the server (the databases involved, the SQL commands to execute, etc.). Each request is labelled with a unique name that is used by the client to identify the desired request. Thus, the client does not need to know anything about the data sources involved; the name of the request, together with some parameters, completely defines the request. The main advantage of this strict separation is that changes to the data structure are possible without modifying the client. After changing the database structure, only the descriptions on the server must be updated – changes to the client-side code are not necessary.

Extending an AIS Client by Modules

This section presents example client modules that have been developed and used for different projects. All modules extend the basic framework and use its different services.

GENERAL-PURPOSE MAPPING MODULES

The module for layer editing shows and hides map layers and changes their symbolization (Figure 7). It allows the user to load additional map layers by selecting them from a list. Currently, three different types of layers are supported: raster images, vector line work, and points. The module relies on the Resource Loader service to load additional map layers. It uses the Map Controller service to integrate the new layers into the map. The module also allows the user to change the graphical appearance of vector layers by specifying colours, line widths, and transparency.

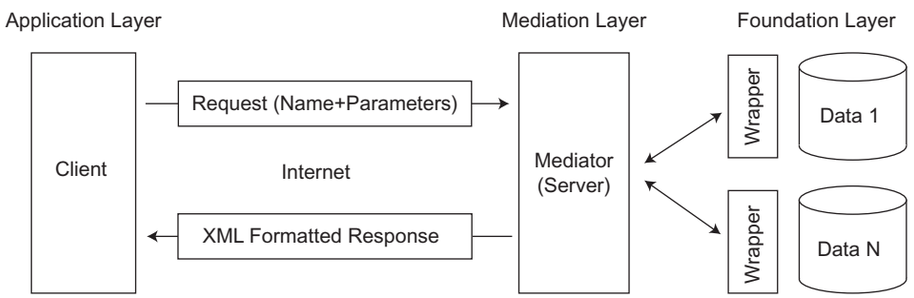

Figure 6. Mediator system wrapping heterogeneous data sources.

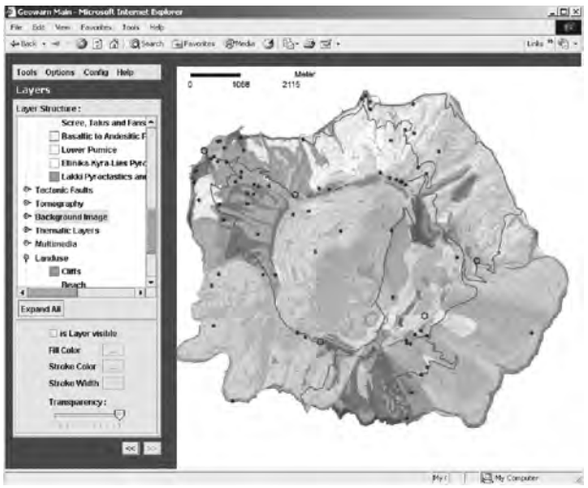

Figure 7. Screenshots of the module for layer editing.

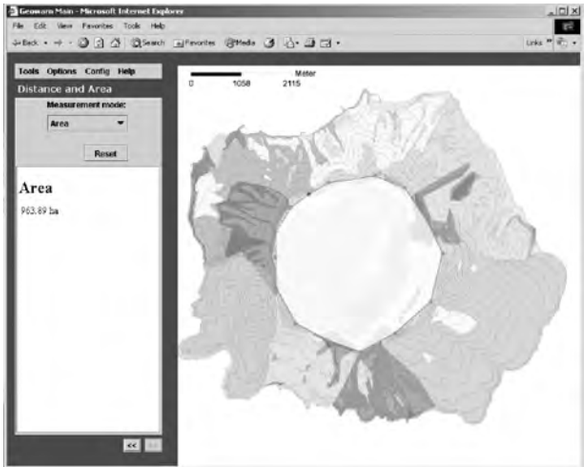

Figure 8. Measurement of an area.

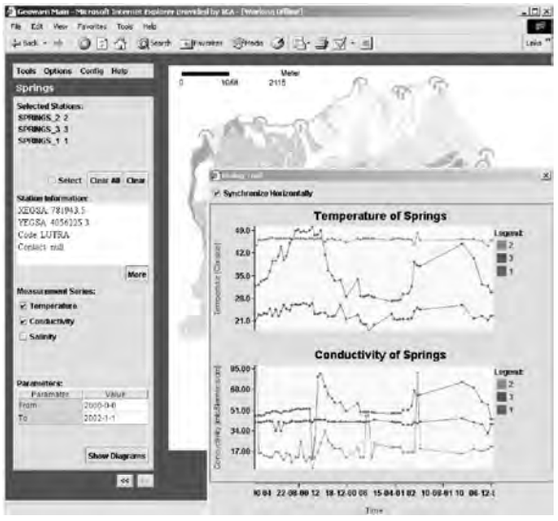

Figure 9. Display of diagrams.

Another elementary module offers zooming and panning capabilities (see Figure 1). Using this module, the user can graphically or numerically enter the new centre of the view and the new map scale. The Navigation Module uses the Map Controller service to scale and re-centre the map.

The user can measure distances and areas on a map using the module shown in Figure 8, which also interacts with the map through the Map Controller service.

The diagrams in Figure 9 show the evolution of temperature and electrical conductivity of three different springs over time. A specialized module produces the diagrams. Parameters passed at initialization determine the module's interface, the data sources, and the type of diagram. The module can thus be easily adapted to different data types, data sources, and database structures.

GEOWARN AIS

The concept of modular Web-based AIS was originally developed for the Geo-Spatial Warning Systems (GEOWARN) project, funded by the European Commission (GEOWARN 2003). The aim of GEOWARN is to develop methods and systems for the surveillance of quiescent but still active volcanoes (Lagios and others 2001). A major part of the project was dedicated to the development of an AIS that embeds all relevant data sets of a volcanic field into a single cartographic system. Uses of the application include scientific analysis, emergency and land use planning, and allowing casual users (civil protection) to make decisions in the event of a seismic or volcanic crisis. GEOWARN uses the volcanic island of Nisyros (in Greece, south of Kos) as a test site that shows high seismic unrest and widespread fumarolic activity. The GEOWARN AIS extends the core framework with a set of custom-tailored modules that facilitate analysis and visualization of diversified geophysical data collected during the project (Chiodini and others 2002; Dietrich and Hurni 2002; Gogu and others 2006; Lagios and others 2001).

The screenshot in Figure 10 shows the module for the analysis of a regularly spaced raster grid. The example shows heat flux in a volcanic caldera. The corresponding module loads and displays a thematic grid using a customizable colour scale. The module handles queries of single grid values and allows for statistical measurements and animated time series of grids.

The module illustrated by Figure 11 extracts profiles from a three-dimensional tomographic model of subsurface geology below Nisyros. The user defines the position of the desired profile graphically on the map. The module then sends the position of the profile to an application server that extracts the profile from a tomographic

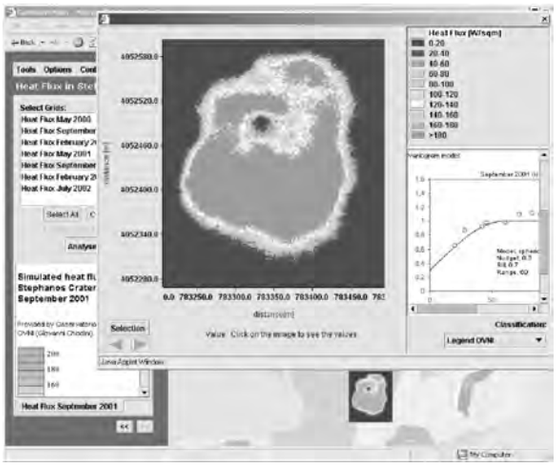

Figure 10. Analysis of raster grid.

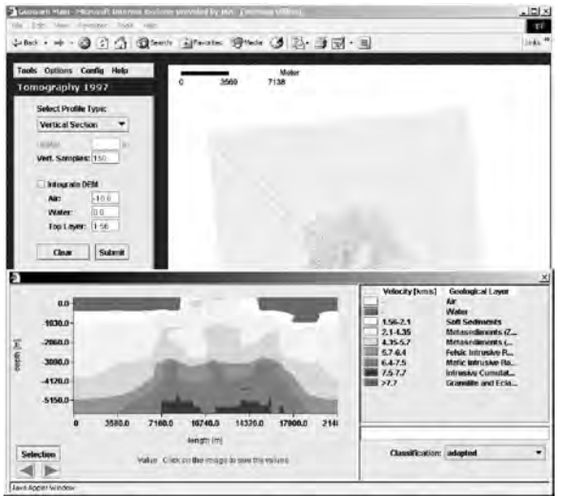

Figure 11. Extraction of 3D model.

voxel model. The server returns the resulting profile as a thematic grid to the active module. The user finally analyses this grid with the functionality described above.

Microseismic events (i.e., seismic events with a magnitude of 3 or less on the Richter scale) are important for the understanding of volcanic processes. A seismic event is defined by a three-dimensional position and a magnitude. The module shown in Figure 12 projects seismic events on a vertical profile, which is then shown in a diagram. The position of the profile can be chosen interactively on the map. The diagram includes only those points lying within a certain distance of the profile.

Conclusion

In addition to the GEOWARN AIS described above, the architecture presented was successfully implemented for

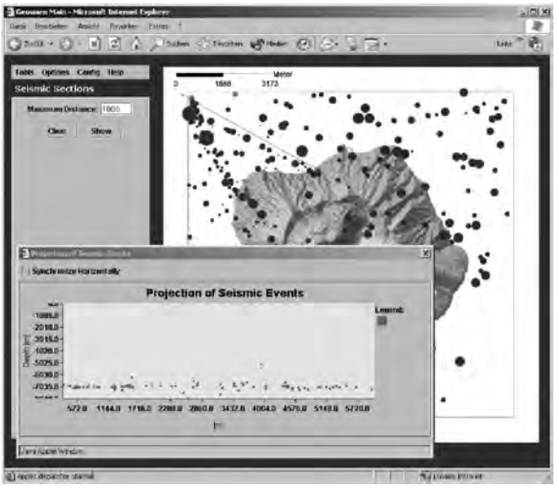

Figure 12. Projection of seismic events on a profile.

AIS of other locations (e.g., an AIS for Campi Flegrei, a volcanic area near Naples, Italy) and in other specialized applications (e.g., a hiking route planner supporting multi-criteria net analysis). Our experience with the modular system was very positive, since it proved to be easily configurable and adaptable to these applications.

On the client side, the modular AIS uses a thin, platform-independent client that is easily scalable to different applications. Its modular architecture allows for rapid integration of new custom-tailored functionality, and its map-viewer architecture can be adapted to future or alternative Web techniques.

The server side uses standard technology (EJB and Java Servlets) that greatly accelerates and simplifies the development of the application server. The separation of the data sources from the rest of the AIS allows multiple data owners to independently update or correct their data, which ensures long-term usability of the AIS.

An entirely Web-based AIS automatically provides the end user with the most recent data and functionalities; complicated installation procedures are unnecessary. The utility of the system is further enhanced by a user-friendly interface and map data that are cartographically pre-treated in order to optimally communicate the information.

Author Information

Bernhard Jenny, Institute of Cartography, ETH Zurich, CH-8093 Zurich, Switzerland. E-mail: jenny@karto.baug.ethz.ch.

Andrea Terribilini, OneOverZero, Müllerstrasse 8, CH-8004 Zurich, Switzerland. E-mail: andrea.terribilini@oneoverzero.net.

Helen Freimark, Institute of Cartography, ETH Zurich, CH-8093 Zurich, Switzerland. E-mail: freimark@karto.baug.ethz.ch.

Radu Gogu, Department of Geotechnical Engineering and Geo-sciences, School of Civil Engineering, Technical University of Catalonia (UPC), 08034 Barcelona, Spain. E-mail: radu.gogu@upc.edu.

Lorenz Hurni, Institute of Cartography, ETH Zurich, CH-8093 Zurich, Switzerland. E-mail: hurni@karto.baug.ethz.ch.

Volker Dietrich, Institute for Mineralogy and Petrography, ETH Zurich, CH-8093 Zurich, Switzerland. E-mail: dietrich@erdw.ethz.ch

References

Adobe Systems Inc. 2006. "Macromedia – Flash Professional 8." Available at http://www.adobe.com/products/flash/flashpro/

Chiodini, G., T. Brombach, S. Caliro, C. Cardellini, L. Marini, and V. Dietrich. 2002. "Geochemical Evidences of an Ongoing Volcanic Unrest at Nisyros Island (Greece)." *Geophysical Research Letters* 29: 16.

Dietrich, V., and L. Hurni. 2002. "GEOWARN ein Frühwarn-Informations-System für Vulkane." *Spektrum der Wissenschaften,* March: 26–28.

ECMA International. 1999. *Standard ECMA-262: ECMAScript Language Specification*, 3rd ed. Available at http://www.ecma-international.org/publications/standards/Ecma-262.htm

Geospatial Warning Systems [GEOWARN]. 2003. "Geowarn – Geospatial Warning Systems – Nisyros Volcano, Greece." Available at http://www.geowarn.ethz.ch/

Gogu, R. C., V. J. Dietrich, B. Jenny, F. M. Schwandner, and L. Hurni. 2006. A Geo-spatial Data Management System for Potentially Active Volcanoes – GEOWARN Project. *Computers and Geosciences* 32: 29–41.

Hurni, L., H.-R. Bär, and R. Sieber. 1999. "The Atlas of Switzerland as an Interactive Multimedia Atlas Information System." In *Multimedia Cartography*, ed. William Cartwright, Michael P. Peterson, and Georg Gartner. Berlin: Springer. 99–112.

Jenny, B., H. Freimark, and A. Terribilini. 2002. "Entwicklung eines kartographischen Internet-Mapservers und eine erste Anwendung in der Geophysik." *Proceedings of GIS/SIT 2002* (CD-ROM). Zurich: ETH.

Lagios, E., V. Dietrich, G. Stavrakakis, I. Parcharidis, V. Stakkas, and S. Vassilopoulou. 2001. "Will Nisyros Volcano (GR) Become Active? Seismic Unrest and Crustal Deformation." *European Geologist* 12: 44–50.

Ormeling, F. 1995. "Atlas Information Systems." *Proceedings of the 17th International Cartographic Conference, Barcelona*, vol. 2: 2127–133.

Schneider, B. 1999. "GIS Functionality in Multimedia Atlases: Spatial Analysis for Everyone." *Proceedings of the 20th International Cartographic Conference, Beijing*, vol. 2: 829–40.

Sun Developer Network [SDN]. 1994–2006. "Java 2D API." Available at http://java.sun.com/products/java-media/2D/index.jsp

van Elzakker, C.P.J.M. 1993. "The Use of Electronic Atlases." In *Proceedings of the Seminar on Electronic Atlases*, ed. I. Klinghammer, L. Zentai, and F. Ormeling. Budapest: Eötvös Loránd University. 145–55.

Web3D Consortium. N.d. "VRML97 and Related Specifications." Available at http://www.web3d.org/x3d/specifications/vrml/index.html

Wiederhold, G. 1992. "Mediators in the Architecture of Future Information Systems." *IEEE Computer* 25/3: 38–49.

World Wide Web Consortium [W3C]. 2006. "Scalable Vector Graphics: XML Graphics for the Web." Available at http://www.w3.org/Graphics/SVG/