Design Requirements for an AJAX and Web-Service Based Generic Internet GIS Client

Edward Nash[1], Peter Korduan[1], Simon Abele[2], Gobe Hobona[3]

[1]Institute for Management of Rural Areas, Rostock University, Germany.
{edward.nash,peter.korduan}@uni-rostock.de

[2]School of Civil Engineering and Geosciences, University of Newcastle-upon-Tyne, UK.
s.j.abele@newcastle.ac.uk

[3]Centre for Geospatial Science, University of Nottingham, UK
gobe.hobona@nottingham.ac.uk

1 INTRODUCTION

The development of spatial data infrastructures (SDIs) is broadening out from the current systems which contain largely only Web Map Service (WMS) interfaces offering visualisations of data to more advanced systems offering access to the actual data via Web Feature Service (WFS) or Web Coverage Service (WCS) interfaces and even the beginnings of web-service based access to continuously-collected sensor data via the Sensor Observation Service (SOS) and geospatial processing functionality over the Web Processing Service (WPS) interface. These geospatial web service interfaces are based on standards by the Open Geospatial Consortium (OGC). A common component of current SDIs is a portal which advertises the available services and provides a simple viewer to allow users to explore the datasets which are available. Whilst more advanced users may wish to bind the services into their own desktop client, many users are likely to be satisfied with the web-based thin-client offered by the portal. The recent growth in AJAX (asynchronous Javascript and XML) mapping applications shows how web-based portals may be made much more dynamic. The great advantage of AJAX is that no browser plug-ins (e.g. Java Applet, Flash, SVG, etc.) are required to provide dynamic interactions on the client side. However, the majority of AJAX mapping applications (e.g. Google Maps, Multimap, Microsoft Virtual Earth, etc.) are proprietary, and based on proprietary web service interfaces, and therefore not suitable for integration in a GDI context. Open AJAX mapping toolkits such as OpenLayers are however rapidly developing, supporting a range of open and proprietary services such as Google Maps, Microsoft Virtual Earth, WMS and even WFS.

The basic principle of AJAX applications is that a `XMLHTTPRequest` object is created using Javascript in the browser. This object represents an XML-based request, whereby the XML DOM (Document Object Model) tree can be manipulated in Javascript to define the request. The request may then be made, with the results being reported to a defined callback method. The asynchronicity means that the request, server-side processing and response are backgrounded, while the user may continue using the client. This gives the application a more reactive and dynamic feel than if the user must wait for the full server response after each action.

What is currently not available is a complete AJAX-based client capable of acting as a web-service based GIS, with the ability to find and bind new services (catalogues, data, processing) as required. Such an application could be configured to act as a portal to a fixed set of services forming a single SDI (e.g. a regional or thematic SDI), or could be used as a flexible thin-client GIS for general use. The COWS (Client for OpenGIS Web Services) project aims to lay the foundations for such a client based on open standards and open-source software. In this paper some initial results regarding the functional requirements, technical restrictions and required system architecture which are generally applicable when designing a generic web-service based internet GIS are presented.

2 REQUIRED FUNCTIONALITY OF A GENERIC CLIENT

In this section we will outline what we consider to be the functional requirements for a generic web-service based internet GIS client. This is necessarily a superset of the functionality that would be required for individual applications of such client software.

Current portal solutions consist typically of three functional modules;

1. a map widget where datasets may be inspected,
2. a layer manager allowing individual layers to be switched on and off and
3. a catalogue browser allowing the user to find available datasets.

Although some current systems also provide simple query tools (measure, query feature information e.g. using WMS `GetFeatureInfo`), no further analysis functionality such as querying based on attributes or more advanced spatial querying or analysis is available. Many systems are also restricted to displaying raster data from WMS services and cannot support the display of vector data or data from other services such as WCS.

In terms of general GIS functionality, we assume that an internet-GIS should eventually be able to replicate the majority if not all of the functionality of today's desktop GIS. We therefore wish to concentrate here on functionality which is additionally required by an internet-based GIS or which must take a significantly different form for an internet GIS to for a standard GIS.

We therefore define the first requirement of a generic client as the ability to display data, which may be raster or vector data, and may come from a variety of remote servers such as WMS, WFS, WCS, SOS or WPS. Both `GET` (key-value pairs) and `POST` (XML) requests should be supported; for e.g. the WPS, not all functionality is available using only `GET` requests. Since further services may be defined in the future, as well as newer versions of existing standards, the support for different data sources should be through a plug-in framework, allowing the range of services supported by one instance to be restricted or extended as required. A further data source which should be considered is the user's local data. The ability to integrate this with data from remote services would elevate a web-based client to being a general-purpose internet-based GIS.

The second functional requirement is the ability to find, bind and remove layers. Such layers may be data layers from a WMS, WFS or WCS, or may be the results of processing operations from a WPS. The ability to find layers should be based on catalogue services, and it should also be possible to find and bind arbitrary catalogue services in order to discover data or processes. In each case, different parameters must be defined in order to bind a layer; these may be the inputs to a process, a filter on the features to be displayed or simply the style to be used to display the layer. The parameters required for each layer type and the most suitable input form should also be handled by plug-ins in the same manner as for the display of layers.

The third requirement we consider here is that of exporting the data displayed in the client. Image-based data referenced by URL could be simply saved using the usual browser functionality (view/save image). Other data such as vector data or raw raster data which has been rendered (e.g. from an initial WCS request) cannot necessarily be saved locally using this mechanism.

The final requirement which we define here is the ability to save a client session, i.e. the layers and view window currently in use.

3 TECHNICAL RESTRICTIONS

The use of AJAX technologies without use of browser plug-ins places a number of technical restrictions upon the client, which impact on its implementation form. The first of these restrictions is that `XMLHTTPRequest` requests can only be made to the domain from which the original page was served (the so-called same-origin or sandbox security model). This only applies to XML-based requests – simple URL-based requests using `GET` (such as for an image from a WMS) may be made directly to third-party servers, although results may not be displayed if clients have disabled display of third-party images. For a portal site which should only communicate with a restricted set of servers the sandbox security model is not a problem, but for a generic client supporting arbitrary servers then a proxy on the local server is required. Having an unrestricted proxy is however a security risk. The solution used by OpenLayers is to allow proxy requests to a fixed set of domains via a server-side python script. Again, for portal-style clients, this is a viable approach, but for a client which should allow users to add arbitrary services, the proxy must also support proxying to these services. A suggested approach is therefore to use a server-side session mechanism to allow a user to add services which may then only be proxied by that user for the duration of their session. The server-side proxy could then also be used to sanitise any content sent from remote services, e.g. to remove JSON (Javascript Object Notation) which may be used to launch cross-site scripting attacks. Although solutions exist, such as Mozilla's signed Javascript, which allow Javascript to operate outside the sandbox, there is currently no cross-browser support for a standard solution, and even signed code may be refused permission by the user (Powell & Schneider, 2004)

A second technical restriction is that browsers cannot directly display images resulting from `POST` requests: the image must be referenced using a URI. For images from WMS services this is not an issue as these may be referenced with a URI, but for e.g. WPS services where not all functionality is available via `GET`, the use of `POST` is unavoidable. Note that this applies only to cases where an image is returned; where XML is returned then this can be rendered by the client. The best solution to this problem would appear to be the use of a server-side proxy which would cache the result of a `POST` request with an image return type and make it available to the client via a `GET` interface. Since the original `POST` operation may be expensive in terms of data transfer and processing time (e.g. a WPS request), it would also be desirable to limit the number of times this request is made. A 'one-shot' cache which would make the results available for only a single access would therefore be unsuitable; a session-based cache allowing the result to be stored until the user closes the session, or a certain amount of time has expired, would be better. This cached image could then be exposed to the client as e.g. a WMS service.

A further technical restriction is the inability of standard browsers to display raster data formats generally used in geospatial applications, e.g. TIFF and GeoTIFF. Although the majority of services should be capable of serving results in a 'pure' graphics format such as JPEG or PNG, a universal internet GIS should be capable of displaying all geospatial formats. Similarly, although browser/Javascript clients are capable of handling and rendering XML-formatted vector data (e.g. GML), the scalability of these solutions is not guaranteed. It may therefore be preferable to have a server-side rendering component. This could be used for rendering formats not supported by browsers as well as either rendering large vector datasets to raster formats or for simplifying the vector data before forwarding it to the client. A WMS or WFS interface could be presented to the client for these cases.

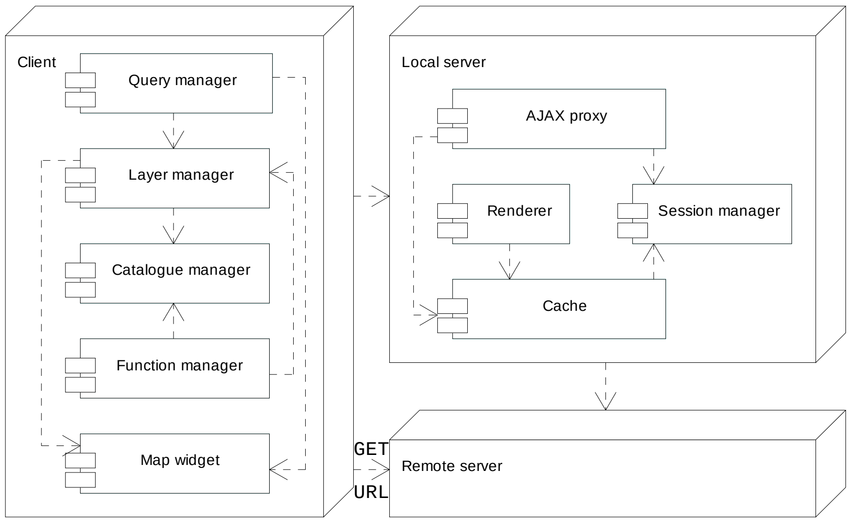

Figure 1: Structure of a generic AJAX GIS web-service client

4 GENERIC CLIENT STRUCTURE

This section will consider the structure required for a web-service based internet GIS based on the requirements and technical limitations previously discussed. Due to the technical limitations, a significant server-side component is necessary to meet the listed requirements, and Sayar et al (2006) reach a similar conclusion in terms of the basic architecture required, although significantly less detailed. Figure 1 shows how such a client may be structured based on a series of interrelated components providing functionality and/or GUI widgets. This structure will now be described in more detail.

Client

We model the client as being composed of a set of interacting components. The central component for the user is the map widget providing the visualisation of the geographic data. Existing components such as OpenLayers may be used for this. Closely coupled to the map widget is a layer manager which would be presented to the user as a legend widget. This layer manager should control which layers are displayed in the map widget and with what styles, with facilities for adding, removing and modifying layers. For performing queries on the data, a query manager is required. This depends on the layer manager for defining what query functionality is available on each layer and the map widget for input of spatial parameters and, where appropriate, display of results. The query manager allows user selections to be converted into the appropriate GetFeature or GetCoverage requests for WFS and WCS respectively. By constructing an XML request within the AJAX client, the user is given a wider range of options as to what to query (e.g. query by bounding box or attribute). A further component on the client-side is the catalogue manager which is responsible for recording the catalogues currently in use and for providing discovery facilities for new catalogues, data and functions. Finally, a component is required to manage the analysis functions provided by WPS or similar processing services. This component must be capable of receiving the required parameters from the user and, if appropriate, adding the resulting dataset to the map widget via the layer manager.

The sandbox security model on the client prevents the direct use of local data in the client. Although display of simple image data may be possible with a workaround, since these may be addressed with a `file:` URL, arbitrary data cannot be directly accessed. In order to incorporate local data it would therefore be necessary to first upload this data to the local server, from where it could be cached and accessed, potentially through a renderer for data which cannot be directly displayed. Similarly for the data defining a user session, this could be stored directly on the server and accessed through a unique user login, or could potentially be saved client-side e.g. using an extended Web Map Context document, which would be uploaded to the local server to re-open the session.

Local Server

The local server requires four components. The AJAX proxy is, as previously discussed, an essential component for communicating with remote servers. The AJAX proxy supports the query manager by parsing the process, feature and coverage descriptions to determine what the permissible inputs are. A session manager is required for security and to allow arbitrary remote servers to be used without running an open proxy. For content-types not supported directly for display in a browser, a renderer component is required. This will not necessarily render the data to a static image, but may simply transform the format into something that can be directly rendered in the browser (e.g. an XML-based format). The renderer may also be used in cases where scalability on the client-side may be problematic, e.g. for large vector datasets or where a low-powered client is being used such as a mobile device. In such cases the renderer should be configurable as to the level of detail and format sent to the client. The final use for the renderer is in making the results of `POST` requests available to the client via a `GET` interface, allowing images to be directly accessed and displayed in the browser.

Coupled to the renderer is a cache to prevent unnecessary calls to external servers and repeated rendering of the same data. For more dynamic data (e.g. sensor data being viewed based on “last 5 minutes” or similar) this cache would need to be configured appropriately or bypassed. The cache may also be used for providing the client with downloads of the ‘raw’ data. The cache is also dependent on the session manager to prevent data being stored beyond its useful lifetime or to prevent users accessing the data being used by others. For persistence of sessions, either a user login system would be required or a session file could be downloaded to the client and subsequently uploaded to the server to re-open the session. In either case a strategy for maintenance of the cache data would be required.

5 DISCUSSION

This paper has presented some design considerations for a generic internet GIS based on web-services and a browser-based AJAX client without plug-ins. The functional requirements of such a system were defined, based on the principle that it should be possible to make available all functionality available in current desktop GIS via such a systen, but that the use of these technologies would impose some extra requirements and restrictions. The main three restrictions are the requirement to have a local server acting as a proxy, imposed by the sandbox security model, the requirement for a renderer for formats which cannot natively be displayed in standard browsers, and the inability of browsers to embed images resulting from `POST` requests. Based on the required functionality and restrictions, a structure for a generic AJAX GIS was developed. This structure is interface and transport-format neutral: although HTTP `GET/POST` to OGC web services is the current standard in SDIs, the use of `WSDL/SOAP` is increasing (e.g. Lemmens et al, 2006), albeit sometimes only as a proxy to an OWS service (e.g. Scholten et al, 2006). The use of SOAP however presents the same problem for browsers as POST in that resulting images cannot be directly displayed; either the SOAP response must contain a URL from which the image may be accessed or a caching proxy would be required.

The requirements of web-service chaining have not been directly discussed in this paper. Particularly in the case of geoprocessing services, the input data may come from a different service, which may itself be a processing service. The recently published OGC WPS 1.0 allows inputs to be specified directly or as a GET (URI) or POST request which will provide the data, theoretically allowing dynamic translucent service chains to be very easily constructed and invoked. The whole chain may be defined in the client using an iterative process starting with the final operation and recursively defining the inputs to this as either local datasets which will be uploaded to the local server either embedded directly in the request or cached and made available to the WPS via a URL proxy, remote resources accessed by URI or the result of a POST request. In future work, a more advanced graphical model-builder could also be implemented using AJAX technologies.

A further use of web-service chaining may be in the use of an external web-service for rendering, i.e. a portrayal service. Although WMS may be considered a portrayal service, it is one which is tightly-coupled to one or more datasets and cannot therefore be used with arbitrary datasets resulting from e.g. processing operations. One option would be to implement a processing service based on the WPS interface, but to enable the display of general geospatial data in a browser, a standardised renderer interface allowing GIS formats such as GeoTIFF to be rendered using a URL/GET-based operation should be developed.

Our study is closely related to on-going efforts by the OpenLayers development community. As already indicated Openlayers is an open source web mapping toolkit based on AJAX technology. Although its primary focus has been on supporting WMS, various developers have extended OpenLayers to support transactional WFS and WPS. OpenLayers, however, is currently implemented as a standalone toolkit meant to be integrated into web-based geospatial applications. Our study proposes to use OpenLayers within web-based GIS that takes the user from the point of searching for a dataset, through to querying and editing the dataset. We therefore envision a client as an application and extension of OpenLayers. In conclusion, the COWS project aims to develop a concept of such a client, whereby existing components such as OpenLayers on the client side and GeoTools, deegree, etc. on the server side should be re-used where possible.

Acknowledgements

The COWS project is funded by the German Academic Exchange Service (DAAD) and the British Council through the Academic Research Collaboration (ARC) programme.

REFERENCES

Lemmens, R., Wytzisk, A., de By, R., Granell, C., Gould, M. & van Oosterom, P., 2006, Integrating Semantic and Syntactic Descriptions to Chain Geographic Services. IEEE Internet Computing 10 (5) 42-52.

Powell, T. & Schneider, F., 2004, JavaScript: The Complete Reference, 2nd edition. McGraw-Hill Osborne, San Francisco. ISBN 0072253576.

Sayar, A., Pierce, M. & Fox, G., 2006, Integrating AJAX Approach into GIS Visualization Web Services. Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006). IEEE Computer Society, New York. ISBN 0769525229.

Scholten, M., Klamma, R. & Kiehle, C., 2006, Evaluating Performance in Spatial Data Infrastructures for Geoprocessing. IEEE Internet Computing 10 (5) 34-41.