An Architecture for Web-based DSS

Huabin Chen [a] , Xiaodong Zhang [b] , Tianhe Chi [a]
[a] Institute of Remote Sensing Applications, Chinese Academy of Sciences
[b] Corresponding Author, School of Information and Electrical Engineering, China Agricultural University
P.O. Box 698, Beijing 100083, China

Abstract: As the web platform continues to mature, we see an increasing number of amazing technologies that take DSS (Decision Support Systems) to new levels of power and usability. By integrating new powerful technologies into DSS, we get higher performance results with additional functionalities. The most recent development capturing the attention of the browser based application developers are Web Service, Struts and AJAX (Asynchronous JavaScript and XML). In this paper we present a generic and performance efficient architeture for integrating Web Services, Struts and AJAX models into the Web-based DSS.

Key-Words: DSS, Web-based, AJAX, Web Services, Struts, MVC

1 Introduction

Information systems researchers and technologists have built and investigated DSS for more than 35 years [1]. Beginning with building model-oriented DSS in the late 1960s, theory developments in the 1970s, and the implementation of financial planning systems and Group DSS in the early and mid 80s, during the mid-1980s we have proposed and implemented Intelligent DSS through combining knowledge system with DSS. Now, the implementation of Web-based DSS in the mid-1990s became active topics and made influence widely.

Web-based DSS mean computerized systems that deliver decision support information or decision support tools to a manager, business analyst, or customer using a "thin-client" web browser like Netscape Navigator or Internet Explorer. The computer server that is hosting the DSS application is linked to the user's computer by a network using the Transmission Control Protocol/Internet Protocol (TCP/IP) and the entire application is implemented using web technologies. A few years before, Web technologies like HTML and JSP had been used to implement any category or type of DSS (communications-driven, model-driven, data-driven, document-driven, knowledge-driven or a hybrid), but the user interface was substandard compared to client-server systems and integration of the applications with database and sophisticated modeling software was limited [2].

Those problems have been overcome by some new web technologies. This article focuses on how to use new web technologies to build Web-based DSS with high retractility, reusability and easy maintaining.

2 Key Web Technologies

Effective decision-making requires the integration of knowledge, data, simulation models, and expert judgment to solve practical problems and provide a scientific basis for decision-making. User-friendly decision support tools are needed to help different stakeholder groups develop, understand, evaluate and share alternative management strategies. The tools should integrate a suite of components consisting of database management systems, other systems like geographic information systems, simulation models, decision models, and user-friendly interfaces that could then be available to different stakeholder groups. Here we propose to adopt some new technologies listed below to improve the system integration, and user-friendly interfaces of Web-based DSS.

2.1 Web Services

Web Services [3] define a platform-independent standard based on XML to communicate within distributed systems. They are loosely coupled and allow short-term cooperation between services. The main protocol defining the kind of communication to a Web Service is SOAP (Simple Object Access Protocol).

2.2 Struts

Struts [4], which utilizes the Model-View-Controller (MVC) model 2, is a free open-source framework for creating Java web applications. The Model represents the business or database code, the View represents the page design code, and the Controller represents the navigational code. Since the framework separate

database code, page design code, and control flow code, larger applications become easy to maintain and extend. Struts works well with nouveau technologies like AJAX and Web Services.

2.2 AJAX

AJAX [5] is an important web development model for the browser based web applications. It uses the JavaScript XMLHttpRequest function to create a tunnel from the client's browser to the server and transmit information back and forth without having to refresh the page. This is meant to increase the web page's interactivity, speed, and usability. AJAX uses XHTML for the data presentation of the view layer, DOM, short for Document Object Model, which dynamically manipulates the presentation, XML for data exchange, and XMLHttpRequest as the exchange engine that ties everything together. The data travels in XML format because it transmits complex data types over clear text. High performance Google Maps [6] is the best-known application which uses this new powerful browser based application model.

Since Struts, AJAX and Web Services are all XML based structures they are able to leverage each others strength. The new Web technologies provide a great opportunity for sharing information and applications with decision makers.

3 Designing and Developing Web-based DSS

Fig.1 helps to illustrate our architecture from a high-level point of view. Based on the three-layer architecture which is the most common web application architecture, we firstly divide the whole system into three parts: the Presentation Layer, the Business Logic Layer and the Data Persistence Layer. And then we adopt new technologies motioned above to improve the system architecture.

The Presentation Layer is basically the GUI (Graphic User Interface) and all of the components associated with the interface. At this layer the data is given a presentation structure that the browser will be able to display. The Business Logic Layer implements the domain specific business processes and rules as well as deals with managing the model computing services and authorization. The Data Persistence Layer comprises all the persistent data stores and the data access middleware. For example, it not only encompasses a relational database, files, or XML documents, but also contains the class library used to visit data such as JDBC (Java Database Connectivity).

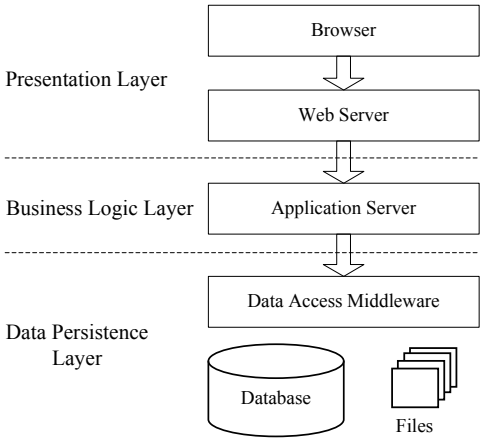

Fig.1 Three-layer architecture

3.1 Web Services in the Business Logic Layer

In the Business Logic Layer, we propose to adopt Web Services technologies (see Fig.2). A new Services Container is added. Also we can name the container as domain services layer. Separating this layer makes our code more reusable and easier to test. With Unit Testing tools, we can easily test code in the domain services layer independently and automatically. When applying this architecture, for easier to manage/test code in this layer and to decouple the web presentation layer from this layer, it is suggested to design services with interfaces.

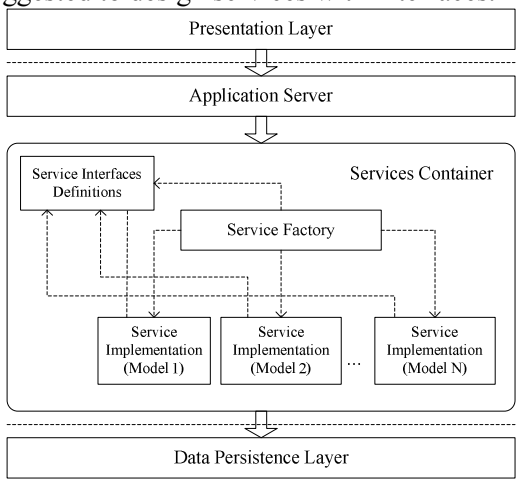

Fig.2 Web Services in the Business Logic Layer

Comparing Fig.2 with Fig.1, we can see the differences are three new components Service Interfaces Definitions, Service Implementations and are Service Factory added. Service Interfaces Definitions are the interfaces of analytical services we want to provide in our DSS. Service Implementations are the codes of implementing the service interfaces. Service Factory is a factory used to produce services, we can pass the service interface to

it and it returns one specific service interface implementation.

As we can see in Figure 2, the web application only depends on the Service Factory and the Service Interfaces Definitions, which means we can use inversion of control to configure services implementations in configuration file (web.xml/ web.config), and Service Factory will discover and load these implementations at runtime and return to web presentation layer. This kind of design decouples the web application layer from the service implementations so that DSS services can be developed synchronously and independently. Extending DSS analytical tools become easier.

3.2 Struts: Connect the Presentation Layer and the Business Logic Layer

The MVC pattern is the most popular design pattern to design and implement the three-layer architecture. Struts which we add in the system framework here is one of the most widely used web MVC frameworks in Java. Fig.3 shows a Controller Layer added between the Presentation Layer and the Business Logic Layer. Also we adopt AJAX in the Presentation Layer.

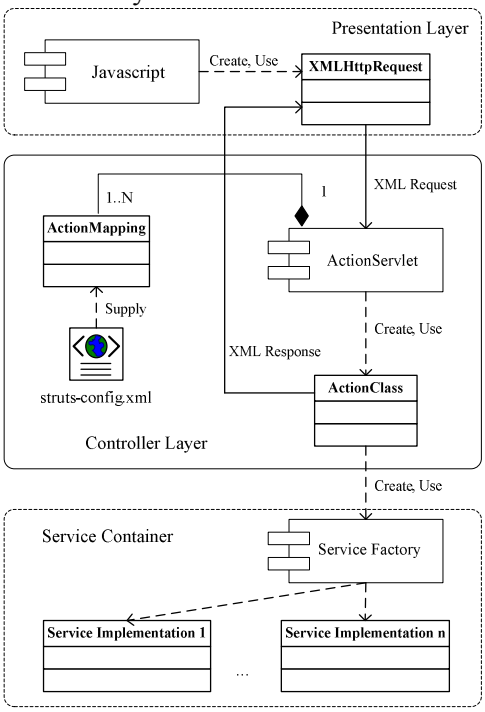

Fig.3 The architecture after Struts added

The Controller Layer is composed of ActionServlet, ActionClass and ActionMapping. ActionServlet, which plays the role of controller, is a servlet that maps events (an event generated by XMLHttpRequest here) to classes. All the requests to the server go through ActionServlet. When an event incomes, the servlet container turns it into an HttpServletRequest. The ActionServlet looks at the incoming event and dispatches the request to an ActionClass according to a configuration file in XML (named struts-config.xml). The struts-config.xml determines what ActionClass the Controller calls. The struts-config.xml configuration information is translated into a set of ActionMapping, which are put into container of ActionMappings (a collection of ActionMapping objects).The ActionMapping contains the knowledge of how a specific event maps to specific Actions. The ActionServlet passes the ActionMapping to the ActionClass via the perform() method.The ActionClass is a model wrapper around the business logic. The purpose of ActionClass is to translate the HttpServletRequest to the business logic. To use the Action, we subclass and overwrite the execute() method.

Each ActionClass maps to a distinct process. The ActionClass calls the relevant methods in the Sevices Container to make use of a Web Service. The Sevices Container gets the required response or an exception if one is raised, and passes it back to the ActionClass. The ActionClass can either process the result and forward the response to the XMLHttpRequest or call the relevant model service to perform further processing.

The model computing service of DSS is implemented as a set of Java classes named Service Implementation in Fig.3 and should not contain any View-related code. Each model service component will offer a set of services, and the components collectively offer a set of common services as well. In the course of processing, the ActionClass can call the required methods in the relevant model service components. These components pass the required data to the model service, and the model service performs any necessary business logic processing and retrieves any necessary data from the Data Persistent Layer.

3.3 AJAX Model: A Rich Client Interface

User interface is important in a Web development environment, and it probably becomes more important because so many users of various levels of sophistication can potentially access some or all DSS capabilities. The three-layer architecture we mentioned above is essential for collaborative or centrally coordinated DSS, but they raise the specter of network latency, with its ability to break the spell of user productivity. Although a general purpose solution to the conflict between the two exists in asynchronous remote event handling, the traditional request-response model of the classic web application is ill suited to benefit from it.

We've found some key characteristics of the browser of DSS different from what traditional Web applications represent. Firstly, the DSS browser hosts an application, not content. Secondly, the DSS server delivers data, not content. Thirdly, the DSS user interacts continuously with the application, and most requests to the server are implicit rather than explicit. Finally, the Presentation Layer's code lib is large, complex, and well structured and we have to take good care of it.

Due to these issues, we adopt AJAX in our Web-based DSS by writing custom JavaScript codes that directly use the XMLHpptRequest protocol's API. AJAX gives expression to a lot of unrealized potential in the web browser technologies. Google and a few other major players are using AJAX to raise the expectations of the general public as to what a web application can do. AJAX isn't a single technology. Rather, it's a collection of four technologies JavaScript, CSS (Cascading Style Sheets), DOM (Document object Model) and XMLHttpRequest that complement one another.

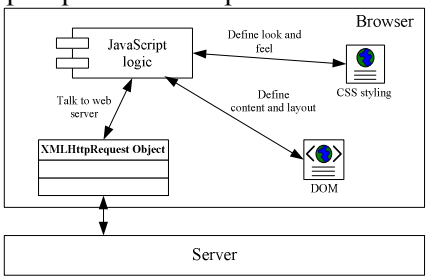

Fig.4 The four main components of AJAX

Our AJAX typically sit at the end of this chain, acting as client only, so we can treat the entire three-layer system as a single black box labeled "Server" for the purposes of our current discussion. Fig.4 shows how the technologies fit together in AJAX. JavaScript defines business rules and program flow. The DOM and CSS allow the application to reorganize its appearance in response to data fetched in the background from the server by the XMLHttpRequest object or its close cousins.

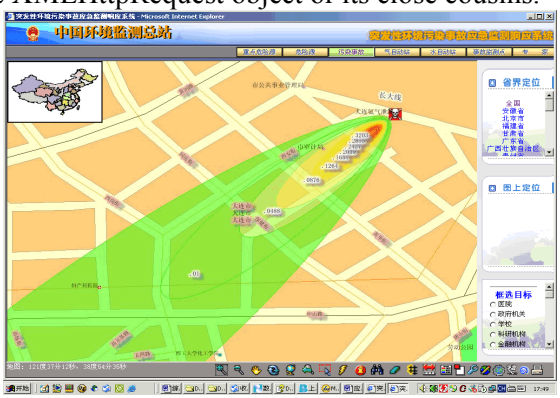

Fig.5 Visualization on the browser by AJAX

Fig.5 illustrates a variety of possibilities for the visualization of DSS user interface in the Presentation Layer. With a map as background, we draw isolines dynamically on the browser to visualize the direction and strength of pollution diffusion between different times.

4 Advantages and Disadvantages

The architecture we proposed above based on MVC allows us to interact with all the features of the application without knowing how the different components interact. Applying the approach to DSS development allows us to take advantage of many well-known merits such as flexibility in the system, easy maintenance and upgrade, reducing the development cycle time by reusing existing components.

Web Services technologies provide a language-neutral, environment-neutral programming model that accelerates application integration inside and outside the DSS. Application integration through Web Services yields flexible loosely coupled DSS. Because Web Services are easily applied as a wrappering technology around existing applications and information technology assets, new model and analytical tools can be deployed quickly and recomposed to address new opportunities. As adoption of Web Services accelerates, the pool of analytical services will grow, fostering development of more dynamic models of just-in-time application integration over the Internet.

By using AJAX in DSS, we can keep all of the visualization functionality in browser area without having to request server a new page. There will be no history pages, since we never left the page. Ajax meets a need in the DSS for richer, more responsive web-based clients that don't need any local installation.

There are also drawbacks in the architecture. First of all, it increases complexity of the system development. Secondly, perfect decoupling of every layer is almost impossible. Some changes which happen in the parameter structure and formats may ignite the changes in the Business Logic Layer and the Presentation Layer. Finally, the AJAX model is not platform-independent. We have to be careful of the coding and implementation differences between different web browsers.

5 Conclusion

Simply making an existing DSS accessible by using a Web browser to stakeholders will often lead to unsatisfactory results. Systematic development

approaches must be explicitly chosen. In our proposed architecture design implementation, we extend the common three-layer architecture and modified the framework of Struts. But we don't change the technologies in the AJAX model and Web Services. By using the same theoretical standards, we can integrate more DSS analytical tools into our existed systems.

6 Acknowledgements

This work is supported by the Institute of Remote Sensing Applications, Chinese Academy of Sciences and school of Information and Electrical Engineering, China Agricultural University.

References:

[1] Power, D.J., A Brief History of Decision Support Systems, *DSSResources.COM, World Wide Web, http://DSSResources.COM/history/dsshistory.html*, version 2.8, 2003

[2] Power, D.J., S. Kaparthi, Buliding Web-based Decision Support Systems, *Studies in Informatics and Control*, Vol.11, No.4, 2002, pp. 291-302.

[3] Gottschalk, Introduction to Web services architecture, *http://www.research.ibm.com/journal/sj/412/gottschalk.pdf*

[4] http://struts.apache.org/

[5] Rob Gonda, AJAX World Magazine Special Feature: What Is AJAX?, *http://cfdj.sys-con.com/read/138966.htm, 2006*

[6] http://maps.google.com