ABSTRACT

This paper investigates and proposes solutions framework for integrating and leveraging existing information sources through web-based decision support. Software architectures for enabling the information service delivery have been studied, particularly from their design and implementation perspectives. The potential of the web-based solutions framework has been analysed at practical levels involving the modern technologies of data communications, SOAP, XML and SUN’s J2EE.

1. Introduction

The use of web-based information delivery service is driven by users needs. Based on the user profile information, data and information services can be delivered and presented intelligently. For example, appropriate levels of service features (available within the software products) are presented to the users based according to their sophistication and skills. This will generate a user oriented decision-making viewpoint dynamically at any stage of the interactive request-delivery process. The intended use of the data influenced by user profiles can assist in determining how the data can best be stored and presented, and serving users effectively. The degree of data access and connectivity that such a system would require can be best delivered by the Web, hence our decision to focus on the development of a web-based decision support system. The above considerations lead us to explore possibilities on the effective use of the Web’s unique potential to meet the new requirements. The information service delivery can be effectively utilised to serve various business needs, such as in a decision-making process, once the associated knowledge is elicited, managed and applied.

2. Architecture Overview

To meet the fundamental requirements, a general architecture has been proposed which is based on the previous work [1] and described in Figure 1. The framework has been designed to interact with a range of the information resources such as database systems, Enterprise Information Systems (EIS) and legacy systems via the web in order to deliver user specific solutions. Users requests are generated through a client application (normally within a user’s work environment) e.g. a client User Interface (UI), as shown in figure 1. The request is transmitted and received by the framework Interface Manager that activates the services offered by the selected back-end components.

The knowledge Management component offers the framework the capability of interacting with the client applications/request intelligently as well as directing client tasks accurately to the responsible targets across the network. The Data Management component will ensure that the framework is capable of dealing with a wide range of external database systems regardless of their locations or formats. The integration of knowledge management and data management delivers the framework the power of processing large volume of information in a real-time manner. Having a powerful business logic processing capability supported by all the required conventional computing services such as data management services, opens up a new solution perspective to real-time transactions delivery via the Web. The framework is based on a java implementation environment.

An important aspect of our framework is its ability to connect geographically dispersed systems. This is illustrated by figure 2. In figure 2, both the organization’s servers and users are geographically distributed. This dispersion can be in either fixed or mobile. For example a laptop on an airplane can access the system as shown by User 4 in Fig 2. As well, authorized users from outside the organization can access those parts of the system to which they have authorization. Similarly, the web based decision support system itself may reside on one sever and access the other parts as required or it may be distributed over several or all of the servers. However this geographical distribution is transparent to the user to whom the system appears logically the same.
Our framework also caters for a federation of organizations of the type depicted in Figures 2. That is, each organization in the federation may have its own geographically dispersed system with each organization’s system linked to the other organizations in a federation. In this federation, the federal web-based decision support system may consist of each organization’s decision support system linked together in a federation or it may be a decision support system designed for the federation and distributed over the servers appropriate for the situation. This ability to connect geographically dispersed systems is facilitated by our framework’s integration with the web. Where there is a need for communication over geographically dispersed locations, the web offers extremely significant advantages – it is cost effective, its geographical coverage is almost omnipresent, support tools and technologies are ubiquitous and use of the browser is almost universally understood and accepted.
3. Related Work

The demand for integration has led to the development of Enterprise Resource Planning (ERP) and Enterprise Application Integration (EAI) systems. An overview of ERP and EAI and their technical challenges can be found in [2] and [3]. To address the integration challenges, Web Services proposes to address the issues of data and systems interoperability as well as process integration. However, technical obstacles remain; mainly in the areas of transaction processing and real-time delivery of information services and these obstacles can directly affect the success of e-business. To tackle these challenges, a knowledge driven Web Services delivery framework was proposed in [4]. However, the work was focusing more on the conceptual issues with little implementation detail supplied. In this paper, we expand previous work by giving a software development perspective, discussing the issues and alternatives required in delivering the proposed framework at the practical level.

4. Deployment Considerations

We have selected SUN’s J2EE as the candidate technology to implement the above configurations. Access to EIS’s and legacy systems is to be via adapters using custom sockets, Java’s JMS technology, custom access libraries or J2EE’s connector interface as appropriate. An overview of the technologies to be used by our framework is shown in Figure 1. As is shown in Figure 1, our decision support system (individual, distributed or federated) is composed of four parts, an interface manager, a knowledge manager, a data manager, and a component connector. These sit in a J2EE application server. While initially our web based decision support system is to be implemented using J2ee, future versions may be able to integrate decision support systems written under various languages. This integration could be achieved by using CORBA orbs or using other connector technologies as they become available (it is anticipated that there will soon be connectors to bridge EJB and Microsoft’s COM [5].

Another pillar of our framework is XML, which is to be the language used to represent data and knowledge in our framework. The choice of XML is natural since it is capable of rich expression, is almost universally accepted and used, is well supported by Java technology and is also supported by a myriad of tools. Naturally, each server in any organization may also be a client of another server in the same organization or another server in another organization in the federation.

5. Data Communications: SOAP and XML

One of J2EE’s underlying technologies is RMI, which is the native J2EE/Java technology for implementing remote procedure (RPC) calls. Both Entity and Session beans implement the remote interface and as such RPC via RMI is natural and relatively easy to implement in a J2ee implementation. An alternative to RPC is messaging where instead of a client invoking a method on a server, the client sends a message to the server and receives a response. Messaging has some advantages over RMI- it is largely language and platform independent and so messaging implemented in the Java/J2ee environment by using Java’s JMS technology could send a message from a Java program to a COBOL program running on a mainframe. As well, messaging is better suited for communication between systems operating on disparate time frames for example where a batch system communicates with a real time system. A third method of implementing remote communication is to implement RPC’s or messaging via the SOAP protocol (the ability to integrate messaging, SOAP and J2EE is quite recent and is enabled by the SOAP with Attachments API for Java (SAAJ) [7]). SOAP uses XML to encode requests and results and works naturally with common internet transport protocols such as HTTP, SMTP and POP3. As well, unlike RMI, SOAP is not a Java only solution and thus its use makes it easier to expand a system to include non–Java applications. A nice SOAP implementation is available from Apache at http://xml.apache.org/soap.

Despite SOAP’s relative slowness, its advantages as an internet protocol using XML make it our mechanism of choice for implementing communication particularly between organizations. The use of RMI would be restricted to local intra-organization communication. Similarly in general, the use of JMS would be restricted to local intra-organization communication. However as discussed below, JMS could also have a role in connecting legacy systems to our framework. The role of the Messaging System module would be to control these communications and protocols.

6. The Connector Architecture

As indicated earlier our framework allows for connection to both EIS’ and legacy systems. As well, as indicated earlier the options for achieving this connection include writing custom sockets and
terminal emulations, using a custom library or using J2EE’s recently developed Connector Architecture. A detailed specification of the connector architecture can be found in [6] with a more general overview provided in [7] and [8]. Where possible, using the connector architecture is our preferred method for connection to both EIS’s and legacy systems. The Connector Architecture has the advantage that it encourages EIS vendors to develop adapters since once developed it can be used by any application server that supports the connector architecture. This should ensure that a wide variety of adapters should soon be available. Indeed most of the major EIS vendors already supply or have indicated a willingness to supply resource adapters for the Connector architecture (the list of contributors to the development of the connector specification JSR 112 include BEA, Bull, Ericsson Infotech, Fujitsu Limited, Hewlett-Packard, IBM, Bahwan Cybertek, Inprise, IONA, MicroFocus, NEON Systems, Inc., Oracle, SAP AG, Siemens, Silverstream Software, Softwired AG, Sun Microsystems, Sybase, Tibco Software Inc, Unisys, WebMethods Corp [6].

As well, many mainframe systems support various type of messaging systems, which can be accessed via JMS. The Connector architecture treats the JMS API provider as a resource adapter with the fact that it is JMS being transparent to end-users. Thus, a large number of legacy systems can also be accessed via the connector architecture.

7. Common Client Interface (CCI)

From the EJB developer’s point of view, the most important part of the Connector API is the Common Client Interface (CCI) which is analogous to JDBC in that just as JDBC provides a standard way of making connections to DBMS’ and creating statements to access the DBMS and retrieving results from the DBMS, so CCI provides a standard way to connect to EIS and to send and receive data from them. Like JDBC, CCI is relatively easy to use as indicated in Figure 7 below. As well, especially important from our framework’s point of view, CCI supports XML thus allowing interaction with EIS using XML. Where connector adapters don’t exist the use of custom libraries such as IBM’s gateway for its CICS system is preferred over the writing of custom sockets. The writing of custom sockets is avoided where possible since mainframe applications are notoriously unfriendly to Unix, Linux or windows applications.

8. Knowledge Management Facilities

The knowledge management module contains the domain knowledge expressed as XML documents. This knowledge representation could encompass knowledge expressed as rules, cases, frames, and scripts etc. all of which are amenable to representation in XML. The inferencing mechanisms which process the knowledge would also reside in this module and would naturally have to be able to process XML. The parsing of documents would be done with either J2EE’s DOM parser or J2EE’s SAX parser as appropriate. The trade off is ease of use versus speed with DOM being easier to use and SAX being faster. In general for the processing of complex knowledge DOM would be preferred since a SAX implementation would require tortuous navigation. An alternative to DOM or SAX is the use of technologies that use java-specific XML tags such as JOX or KBML. However we forgo these technologies since it is unlikely that you would find many partners willing to use java-specific tags particularly if you later want to extend the system to non-java users.

9. Support for Interoperability

The use of EJB in data management is well understood and our framework does not add anything unusual to the field. As is standard practice, we use entity beans to represent persistent objects and session beans to implement the business logic and interact with the session beans. We prefer that in the entity beans persistence be bean-managed (BMP) rather than container managed (CMP). This is because while writing CMP code is easier than writing BMP; the quality of the CMP is directly dependant on the persistence manager implementation. As well, communication problems may arise in a federated system, with different partners using different CMP’s.

The framework is likely to contain a very large number of beans (components). Determining which combination of components is required to satisfy a particular application is the role of component connector in our system. This part of our framework will make use Java’s reflection package and factory pattern to dynamically load classes as required. Figure 8 below, gives an example of an implementation that accepts a class name as an argument and then runs it.

10. Interface Manager

The Interface Manager module will, as its name implies manage interfaces between the users and
the system. As would be expected for a web-based system, this part of our framework uses JSP and XSL. XSL of course is a language that is used to translate XML from one format to another and can be used to process XML data and render a web page. “A JSP page is a text document that contains two types of text: static template data, which can be expressed in any text-based format, such as HTML, SVG, WML, and XML, and JSP elements, which construct dynamic content [7]. That is JSP can render a web page and in addition can also fetch data which XSL cannot. However to render a document expressed in XML, JSP needs to apply an XSL style sheet to the XML document. Thus for an XML based system such as ours, the two technologies are complementary.

11. Evaluation and Applications

So far, this paper has discussed implementation issues in relation to our frameworks components. Here we would like to briefly discuss the current state of the framework and to take another look from application perspectives. Because of our framework’s flexible knowledge handling capability, the ease of incorporating problem-solving information into our framework’s components and its adequate data management support, software components can be re-configured or re-built on demand to suit different application needs. One example use of this framework capability is the deployment of multi-agent team to support knowledge management infrastructure [9]. These agents can be tailored to perform the designated tasks across the web driven by data communication facilities and agent knowledge bases that are supported by the framework. The knowledge management component of the framework has been delivered to the computer labs for teaching purposes. At the present stage, we are focussing more on the J2EE support of our implementation tasks. Preliminary study on Microsoft .NET connection and implementation strategy is currently underway.

12. Summary and Conclusion

In this paper, we have proposed web-based decision support architecture and outlined its general aims and components functionality. We examined the framework specifically from implementation perspectives, aiming to deliver cost-effective and robust solutions. The concepts of SUN’s J2EE and its facilities [10] were extensively borrowed and used within the framework. We aim to use this paper to help practitioners such as application developers to focus on the appropriate software development issues. The framework has been designed to work with existing information resources (such as database systems and software application packages). Currently, several lightweight client applications are being developed to support the demands coming from different domain areas. The framework is gearing up for web-based decision support with industrial strength.

13. References:


