WEB SERVICES-BASED INTEGRATION OF HETEROGENEOUS DATABASES USING A MULTI-AGENT SYSTEM

Hazem T. El-Khatib,
College of Information Technology, United Arab Emirates University,
Al Ain, P.O.Box 17555, United Arab Emirates.

M. Howard Williams, David H. Marwick and Lachlan M. Mackinnon
School of Math and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, UK

ABSTRACT

An important application area in which software agents have an essential role to play is that of providing user access to a collection of heterogeneous distributed databases. This paper describes a general approach, which uses a multi-agents system based on Web services and common object request broker architecture (CORBA) to retrieve information from a set of heterogeneous distributed databases and consolidate it, taking account of the syntactic and semantic differences between data sources. The system is dynamic in that the data sources accessed by the system can change with time. The ability of software agents to negotiate and co-operate with each other make them ideal candidates for this kind of system. Web services can move software agents to the Internet. The role of CORBA in such a system has been investigated. Initially CORBA was used to enable software agents to access each others' services - although this limits the flexibility of the system. A better solution is based on the use of Web services while CORBA is used as the technology to implement the business logic for the software agent, and will be exposed to the Internet using Web services.

KEYWORDS

Software Agent, Web Services, Heterogeneous Databases.

1. INTRODUCTION

Distributed database development has been an important field ever since computing and data storage moved from monolithic architectures and encapsulated jobs on centralized mainframe computers to client/server architectures on peer-networked workstations and subsequently to n-tier architectures. The distribution of databases and services across multiple workstations evolved as a way to scale and organize the data center using a variety of approaches to communications and low-level facilities for integration with arbitrary applications. The information systems strategies of enterprises moved from database centric to the perspective of information flow centric leading to a new generation of business services. However, this approach requires those business logics scattered over the Web to be made accessible in the form of services available throughout the Internet.

This paper describes the problem of integrating information from heterogeneous distributed databases in response to a user query. This problem is complex and it has attracted a considerable amount of attention over the years and has been investigated by a number of researchers.

Various papers have been published describing a variety of different approaches, which have been developed to handle this problem, although aspects of the problem remain. Surveys and comparisons of systems developed to handle this problem can be found in [Thomas, Thompson, et al.] [Litwin, Mark, et al]. This paper describes an on-going system that integrates and retrieves information from a collection of databases using Web services that are encapsulated within software agents. It distributes the task of resolving heterogeneity between autonomous and co-operating agents. Moving knowledge from a centralised
knowledge base system and distributing it over the network and using software agents to access this knowledge increases the number of databases that can be handled by the system significantly. Software agents help in solving the heterogeneity between data sources by dividing the problem between different types of agents. Furthermore, Web Services enable software agents to be accessed by anyone, anywhere, using any type of hardware. They act as glue that allows software agents to be pulled together across networks. Software agents that offer stock quotes, weather reports and credit card validation could be offered as Web services.

A major challenge in developing a system to integrate information from heterogeneous distributed databases is to resolve the heterogeneity that may exist between different databases and to hide this heterogeneity from users. To assist in handling this problem a number of different ways of classifying heterogeneity have been put forward by different authors. The classification adopted for this system [El-Khatib, at al.] analyses heterogeneities in terms of one or a combination of the following:

- Naming heterogeneity – In this case the same object is referred to by different names in different databases (naming synonyms) or the same name is used to refer to different objects (naming homonyms).
- Relational structure heterogeneity - The composition of attributes into composite structures varies.
- Value heterogeneity - The way values are represented is different in different databases.
- Semantic heterogeneity – This involves differences in meaning, e.g., differences in what the data represents or the context in which the data has been captured in different databases.
- Data model heterogeneity – In this case the data model itself is the issue and transformations between data models and differences between them are relevant.
- Timing heterogeneity – The structure of a database, the representation of attributes and/or the way values are represented, may change over time.

2. SOFTWARE AGENTS

The notion of an agent is becoming increasingly important due to its natural metaphor for conceptualising, designing and building complex distributed applications, and its generality, flexibility, modularity, and ability to take advantage of distributed resources [Hayes] [Wooldridge, etc.]. Agents help users in different ways that users need not even be aware of: they hide complexity of difficult tasks, they perform tasks on the user’s behalf, they can train or teach users, they help different users collaborate, and they monitor events and procedures [Maes] [Norman]. Riecken [Riecken] asserted “The basic idea of agent research is to develop software systems, which engage and help all types of end users”. Agents improve system robustness (when one agent is destroyed, others can still carry out the task), and assist humans by reducing their work and information loads [Hayes].

There is no commonly agreed-upon definition of exactly what an agent is [Nwana and Ndumu] [Jennings and Wooldridge], partly because the term agent is both a technical concept and a metaphor and it has been used in several senses. Nwana [Nwana] gave two reasons why it is difficult to define exactly what agents are - “Firstly, agent researchers do not own this term in the same way as fuzzy logicians/Al researchers. Secondly, even within the software fraternity, the word agent is really an umbrella term for a heterogeneous body of research and development”. Several words have been used by researchers to describe their agents, such as intelligent agents, intelligent interfaces, adaptive interfaces, knowbots, knobots, softbots, userbots, taskbots, personal agents and network agents [Riecken] [Nwana]. Hayes [Hayes] noted that many elaborately stated that an agent is an entity that is capable of carrying out goals, and is part of a larger community of agents that co-operate with each other. Many common attributes (e.g. autonomy, intelligence, mobility, collaboration) are beginning to emerge through the literature [Franklin and Graesser]. Minsky [Minsky] defined an agent as a person who acts on another’s behalf or a person who works for or manages an agency - “The idea was to use the word ‘agent’ when you want to refer to a machine that accomplishes something without your needing to know how it works. You call it an agent when you want to treat it as a black box”. Russell and Norvig [Russell and Norvig] defined an agent as “anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors”. Smith et al. [Smith et al], defined an agent as “a persistent software entity dedicated to a specific purpose. ‘Persistent’ distinguishes
agents from subroutines; agents have their own ideas about how to accomplish tasks, their own agendas. ‘Special purpose’ distinguishes them from entire multifunction applications; agents are typically much smaller”. Wooldridge and Jennings [Wooldridge and Jennings, Jennings and Wooldridge [Jennings and Wooldridge] defined an agent as “a self-contained problem solving system capable of autonomous, reactive, proactive and social behaviour”.

As these definitions make clear, there is no general agreement as to what constitutes an agent. However, for the purposes of this paper, a software agent is presented as a persistent well defined application solution that is self contained, has a clear purpose, is intended to be shared, cooperative, loosely coupled, discoverable, has well defined interfaces, accepts well-defined input and delivers well-defined output, and is linked to specific processes.

3. WEB SERVICES

The approach used to support communication between the agents in the system described here is based on the use of the common object request broker architecture (CORBA) [Henning and Vinoski]. CORBA is an open, vendor-independent object-oriented client-server architecture and infrastructure for distributed object technology. While CORBA is widely used as the basis for many mission-critical software applications in a number of areas such as telecommunications, finance, ecommerce, and healthcare, it requires a well-defined architecture with stable interfaces (Interface Definition Language) between components. This requirement can be a problem. It is difficult to create the technical agreement and coordination needed to build a distributed object system. Add to this the fact that CORBA does not operate effectively over the Web and developers often face the significant challenge of Web-enabling these systems.

Web services can move distributed applications to the Internet. Different vendors agree that Web services will solve the front end problem with CORBA. CORBA can be seen as the building blocks of a robust, scalable, highly available service within an enterprise while Web services are highly accessible entry points to those systems [Conway]. Web Services offer a great opportunity to extend CORBA systems. It is believed that Web services will become the next wave in the evolution of Internet application development and is currently the most widely discussed emerging technology. The International Data Group (IDC) estimated that the total software, hardware and services opportunity derived from Web services will raise from $1.6 billion in 2004 to $34 billion by 2007.

Web Services standards technologies are based on the open standards service-oriented architecture recommended by the World Wide Web Consortium (W3C). The key Web Services standards are Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery and Integration (UDDI), which are all based on Extensible Markup Language (XML) [Newcomer]. These technologies work together to provide a Web service model with important functionality as shown in Figure 1. Web service and the backend system (which implements the business logic) represent a software agent in this system. The WSDL is used to create a Web Service that defines its interfaces and invocation methods, followed by publishing to one or more Intranet or Internet repositories for potential users to locate, where UDDI is used to enable organizations to list their Web services in an Internet directory for other organizations to discover and use, and the SOAP messages are used to invoke and consume the web service.

![Figure 1. Web Service](image-url)
4. SYSTEM ARCHITECTURE

This section gives a brief overview of the architecture of the system and describes the roles of agents in the Query Layer, the Answer Layer, and the Information Finder Layer as shown in Figure 2. The software agent which encapsulates the business logic was implemented using Java and CORBA. This architecture takes a user query and breaks it down into a set of sub-queries, which can be sent to appropriate databases. The returned data is processed to resolve heterogeneities and assembled into a single answer. In order for the system to be flexible, scalable, easy to deploy and easy to maintain, a multi-layer architecture was adopted. This breaks down the application into several layers. The system architecture consists of six layers, the Presentation Layer, Query Layer, Answer Layer, Information Finder Layer, Metadata Layer and Database Layer.

4.1 The Presentation Layer

The Presentation Layer is the interface with the user. It accepts the user query through some interaction styles (for example, commands, menu, form fill-in, etc.) and presents the results to the user. At this stage a very simple version has been implemented with a more complete version planned for the next phase for this system; however references to other sources such as [Shneiderman] have been included.

4.2 The Query Layer

The Query Layer consists of a set of software agents. These agents accept the user query from the Presentation Layer and map the query concepts to the ontology concepts. After that, the ontology concepts are mapped to the data source concepts to locate a suitable data source to answer the query. The Query Layer produces a set of sub-queries to be sent to different underlying databases. When the data is returned from these databases it is processed to resolve heterogeneities and assemble a final answer. This is done in the Answer Layer. To aid in this process an internal structure is constructed in the Query Layer, referred to as the Intensional Structure [MacKinnon et al]. This structure is used by the Answer Layer’s agents to rebuild the semantic information lost in the change from intensional to logical queries in the Query Layer. The Intensional Structure contains information about the semantics of the original query. This information is used at different stages of the answer construction process.
4.3 The Answer Layer

This layer is responsible for constructing answers from data returned from the data sources with the aid of the Intensional Structure, and for resolving any conflicts that may occur in the result. It receives the answer from the Information Finder Layer, builds suitable answer storage, and resolves conflicts by communicating with the Information Finder Layer, which provides information on the data sources.

4.4 The Information Finder Layer

This layer consists of a set of software agents responsible for finding and searching for suitable data sources that can be used to satisfy the user query and for helping to resolve conflicts between data sources. It communicates with the metadata for these data sources and uses ontologies to map from user query concepts to database concepts. The External Data Access Agent (EDA) is a Mediator, i.e. a process situated between “provider” processes and “consumer” processes and which “performs services such as integrating information from several sources; translating queries or replies” [Wiederhold]. It acts as a server to the Sub-query Construction Agent and the Data Handling Agent, and as a client to the underlying databases. It can also be described as an Information Agent [Wooldridge and Jennings] [Mckay, et al]. The system can have one or more EDA Agents dedicated to different domains.

The Sub-query Construction Agent in the Query Layer passes the query to the EDA Agent. The EDA Agent validates it, and informs the Sub-query Construction Agent of the result. The query is then separated into individual sub-queries (if it consists of a set of individual queries). The EDA Agent translates each sub-query into SQL, and contacts the database. The EDA Agent links to each target database using JDBC. The answer is then collected from the database. Once all sub-queries are complete the EDA Agent waits for the Data Handling Agent to request the subquery answer number. When this occurs, the answer is pulled to the Data Handling Agent for further operations to be performed on it.

4.5 The Metadata Layer

To avoid the problem of centralised information administration, a Metadata Layer is used. The Metadata Layer consists of a set of eXtensible Markup Language (XML) [Sagar] documents, which store metadata about the associated databases. This metadata is used to assist the agents in resolving the heterogeneity between different databases. The metadata provides the system with information about the semantics of databases. It helps in mapping between the databases’ attribute names and the attribute names in ontologies.

4.6 The Database Layer

This layer consists of the underlying databases used in the system. The main focus at this stage is on relational databases only. The communication between this layer and the Information Finder Layer takes place using Java Database Connectivity (JDBC) [Cornelius].

5. CONCLUSION

This paper describes an agent-based architecture for accessing and integrating data from heterogeneous distributed databases which has been implemented using Internet technologies. This improved architecture enables the system to handle a much larger number of databases and for databases to be added or removed with little effort. Furthermore, the use of agents helps in resolving the heterogeneity between data sources by dividing the problem between different types of agents. The system is currently undergoing redesign and re-implementation using Web services technology. Web services are Internet-based applications that provide standard interfaces and communication protocols for efficient and effective integration of software agents.
The agents have been developed in Java and communication between them is effected using CORBA. Web services could be used to provide a layer of abstraction above CORBA. Currently, Web services are being used as a way to Web-enable access to CORBA systems [El Khatib].

REFERENCES


