ABSTRACT

DIGITAL LIBRARIES AND MIDDLEWARE ARCHITECTURE

by
Muhammad Umar Qasim

Digital libraries deliver personalized knowledge directly to their users, without being restricted to the contents of a physical library. In a digital library information from any online source can be managed and shared, making more knowledge available to users than before. The information sharing is achieved by integrating many autonomous heterogeneous systems available. The challenge is to provide users with the ability to transparently access digital library contents in spite of the heterogeneity among the information sources.

Research communities have proposed several approaches to accomplish the system integration in digital libraries. In this thesis, the working of currently employed approaches was assessed and a new ontology based approach is proposed. This approach utilizes the semantic web enables web services to implement the middleware for digital library integration. Ontology based digital library integration will provide a machine processable mechanism and will overcome the shortcomings of earlier approaches.

DIGITAL LIBRARIES AND MIDDLEWARE ARCHITECTURE

by
Muhammad Umar Qasim

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Information Systems

Department of Information Systems

January 2005

APPROVAL PAGE

DIGITAL LIBRARIES AND MIDDLEWARE ARCHITECTURE

Muhammad Umar Qasim

Dr. Michael Bieber, Thesis Advisor Date
Associate Professor of Information Systems, NJIT

Dr. Stephane Gagnon, Committee Member Date
Assistant Professor of Management, NJIT

Dr. Vincent Oria , Committee Member Date
Assistant Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author:	Muhammad Umar Qasim
Degree:	Master of Science
Date:	January 2005

Undergraduate and Graduate Education:

- Master of Science in Information Systems,
  New Jersey Institute of Technology, Newark, NJ, 2004
- Master of Business Administration,
  Hamdard University, Karachi, Pakistan, 1999
- Bachelor of Computer Science,
  Hamdard University, Karachi, Pakistan, 1998

Major: Information Systems

This thesis is dedicated to my beloved family

ACKNOWLEDGMENT

With a deep sense of appreciation, I would like to express my sincere thanks to Dr. Michael Bieber for his immense help in planning and executing this work. His support, encouragement and reassurance are greatly acknowledged. His valuable suggestions guided me in completing this work.

My sincere thanks to Dr. Vincent Oria and Dr. Stephane Gagnon for reviewing this work and providing me important suggestions. Special thanks are due to Anirban Bhaumik for his assistance in this work.

I wish I would never forget the company I had from my friends at my home. I am thankful to Mumtaz, Qazi, Umar, Akhtar and Nadeem for their countless cooperation, help and encouragement.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of this paper is to explore the various middleware implementation approaches for the system integration in the context of digital library integration. Often information sources have been designed independently for autonomous applications, so they may present several kinds of heterogeneity making the integration complicated. The approaches discussed in this paper present architecture to overcome the problems in the digital library integration. A detailed architecture of one approach is presented, which is being implemented at New Jersey Institute of Technology. A new approach for the digital library integration has been suggested for the future implementation. This approach is based on the Semantic web and ontologies.

1.2 Background Information

Information is power, and digital libraries are built to provide a unified infrastructure for supporting the creation of information sources, facilitating the movement of information across global networks and allowing effective and efficient interaction among knowledge producers, librarians, and information seekers (Adam et al. 2000).

Digital libraries are becoming widely accepted as sources for knowledge in all the areas such as science, engineering, education and business. It is being expected that the move from printed documents to digital documents will soon be ubiquitous and digital libraries will become integral and critical infrastructure for the nations around the world.

A digital library is a collection of objects stored and maintained by multiple information sources, including databases, file systems, email systems, the web, and other formats. These information sources are diverse and dynamic in nature. For these reasons digital libraries have countless technical and engineering issues associated with connecting together these networks, databases, and other computer-based systems.

A fundamental issue for digital libraries is interoperability, the capability of digital libraries to exchange and share documents, queries, and services (Birmingham et al.). Interoperability is the mechanism through which different systems work together and exchange data. Interoperability between different systems is achieved by using common standards and specifications. Interoperateablity could also be viewed as an integration of multiple systems. System Integration is the process by which multiple software modules are made to cooperate (Barret et al., 1996). System integration in digital libraries is the ability to generate a virtual view on many different library components without changing their autonomy. Therefore the challenge is to provide users with the ability to seamlessly and transparently access digital library objects in spite of the diversity and vitality among the information sources and the varied nature of the objects. To accomplish this, an effective system integration methodology must be adopted.

The goal of data integration in digital library network is to provide users with a uniform interface to access, relate, and combine data stored in multiple, autonomous, and possibly heterogeneous information sources (Fahmi et al., 2001). Nowadays, a central topic in database science is the need of an integrated access to large amounts of data provided by various information sources whose contents are strictly related. Although there is a continuous and dramatic growth of digital libraries and information sources, however this growth has made the task of finding, extracting and aggregating the relevant information harder. This is because most of the information systems underlying digital libraries are physically distributed, heterogeneous in the way how information is stored, organized and managed, and comprise heterogeneous software and hardware platforms on which they reside. Additionally, they are autonomous in the sense that the content and format of data are determined by the organization owning the data, not by the user (Ibrahim et al., 2001).

We start with the introductory Chapter 2 about the system integration and way of implementing it through middleware approach. In Chapter 3 we will explain different middleware approaches that are currently being used to implement the digital library integration. One of these approaches will be discussed in detail in the Section 4. This is approach is used at New Jersey Institute of Technology to implement digital library integration. In Chapter 5 a new ontologies based approach has been proposed for the digital library integration and its architecture has been discussed. We conclude and suggest the future work in Chapter 6.

CHAPTER 2

SYSTEM INTEGRATION

2.1 System Integration Overview

System integration is the progressive linking and testing of system components to merge their functional and technical characteristics into a comprehensive, interoperable system. System integration allows data existing on different systems to be shared or accessed across functional or system boundaries. System integration is the process through which a number of products and services are specified and assembled into a complete system that will achieve the intended functionality (Sage, 1992). The goal of system integration is to utilize various autonomous systems in concert so that they support the organizational goal, by providing an integrated set of data and services.

System integration is one of the main reasons for the innovation in the software industry nowadays. Isolated applications are being left behind. Currently new systems are designed to be integrated. But this integration has a lot of challenges and complexities. We believe that in the future complexity will be reduced, but the demand for engineering competence in design, development, integration and maintenance of complex systems will increase.

Nowadays, a central topic in database science is the need of an integrated access to large amounts of data provided by various information sources whose contents are strictly related. Often information sources have been designed independently for autonomous applications, so they may present several kinds of heterogeneity (Greco et al., 2002). Many practitioners are not able to repetitively carry out systems integration across different systems and vendor claims on integration packages are not always true (Nilsonn et al., 1990).

The candidate systems for integration can be considered to be composed of three architectural layers: business architecture layer, application architecture layer and technology architecture layer. The business architecture layer defines organizational structure and workflows. The application architecture layer defines the implementation of business logic in enterprise applications. The technology architecture layer defines the information and communication infrastructure (Hasselbring, 2000). System integration problem could be seen in three-dimensions with each dimension having its own objectives (Sikora et al., 1998).

1. Integration of heterogeneous information systems to facilitate the flow of data and make the overall system more robust.
2. Integration of business processes to improve performance.
3. Integration of subsystems into well-coordinated, networked system.

Traditional systems integration is concerned with integration at the architecture and technology layer of heterogeneous systems. This represents system integration solutions commonly referred to as enterprise application integration or middleware technologies. The literature is not specific about the clear distinction between the two and many times both terms are used interchangeably.

Software vendors provide various solutions for enterprise application integration, based either on metadata-, process-, service-, or portal-oriented integration technologies (Linthicum, 2004). In this report, we focus on the Service Oriented Architecture (SOA) as the primary middleware solution (Papazoglou & Georgakopoulos, 2003).

2.2 Middleware

Middleware services sit in a layer above the operating system and network software but below the industry specific applications. They provide standard programming interfaces and protocols that mask the complexity of networks and lower level protocols (Bernstein, 1996). Middleware facilitates the communication and coordination of components that are distributed across several networked hosts. It aims at providing application engineers with high-level primitives that simplify distributed system construction (Emmerich, 2000). Figure 2.1 shows how a digital library interacts with the sources of information through middleware. Middleware is a layer between database resources and the client applications like digital library applications that want to use the database resources. Middleware programs provide messaging services so that different applications can communicate.

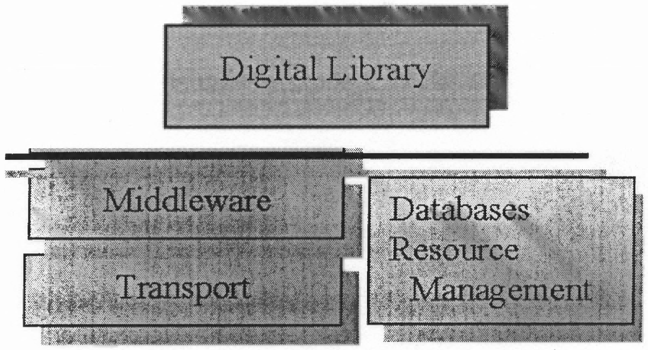

Figure 2.1 Digital library integration architecture.

The main purpose of middleware services is to help solve many application connectivity and interoperability problems. The objective is seamless, large scale integration of heterogeneous components that will provide digital libraries the capability to communicate with heterogeneous sources.

Middleware can take on the following different forms (Bray).

- Transaction processing monitors provide tools and an environment for developing and deploying distributed applications.
- Remote Procedure Call enables the logic of an application to be distributed across the network. Program logic on remote systems can be executed as simply as calling a local routine.
- Message-Oriented Middleware provides program-to-program data exchange, enabling the creation of distributed applications. MOM is analogous to email in the sense it is asynchronous and requires the recipients of messages to interpret their meaning and to take appropriate action.
- Object Request Brokers enables the objects that comprise an application to be distributed and shared across heterogeneous networks.

For the implementation of Digital Libraries we will discuss different Middleware software architectures which are currently being used in the industry and academia. We will also discuss one of these approaches in detail which is being implemented at New Jersey Institute of Technology's DLSI (Digital Library Service Integration) project. Also in the Section 5 of this paper introduces a new approach which can provide effective and efficient middleware architecture. (Adam et al. 2000) in his paper described three approaches, Object Request Brokers (e.g. CORBA), Mediated, and Agent-based, used for purpose of System Integration in the Digital Libraries. These three approaches are not orthogonal in the sense that a mediator approach may also use ORB architecture and an agent-based may use mediators.

CHAPTER 3

DIGITAL LIBRARY INTEGRATION APPROACHES

Middleware used for Digital Libraries is currently developed on the approaches described in this section. These approaches provide the architecture to access the information of heterogeneous nature.

3.1 ORB Approach

Nowadays software integration projects are addressed by using object-based Web technologies. Object Request Broker (ORB) approach incorporates a standards-based framework to support the integration of many software applications. Architectures such as Common Object Request Broker Architecture (CORBA), Component Object Model (COM) and Enterprise JavaBeans (EJBs) support this approach. The utilization of such technologies enables the system to quickly evolve into an assembly of functional components, managed by an inter-object communications protocol and extending to the boundaries of the Internet.

In ORB approach the client object must have an object reference to the server. The ORB approach is enjoying a rapid expansion due to three factors:

- A more rigorous formalization of the approach.
- A greater ease in mastering the development of system integration.
- The rapid emergence of new technologies such as J2EE and .Net, which integrate fundamentally the idea of components.

CORBA is one of the architectures being used to implement this approach. CORBA is an acronym for Common Object Request Broker Architecture. The Object Management Group (OMG) was formed in 1989 to develop standards for application development within heterogeneous environments.

CORBA enables the creation of distributed object architectures. CORBA consists of ORB Core, Interface Definition Language (IDL), Stubs, Skeletons, and others. ORB Core is responsible for delivering requests to object implementation and responses from objects to the client requesting the service. CORBA provides the IDL as a mechanism for specifying the interface to an object. If a developer desires to make use of a service, the IDL specification is the only information required. The specification is compiled by an IDL processor that creates a stub which is linked into the client. The IDL compiler also generates a skeleton file that is used to connect the implementation of a service into CORBA's distributed object architecture. Figure 3.1 shows the CORBA architecture.

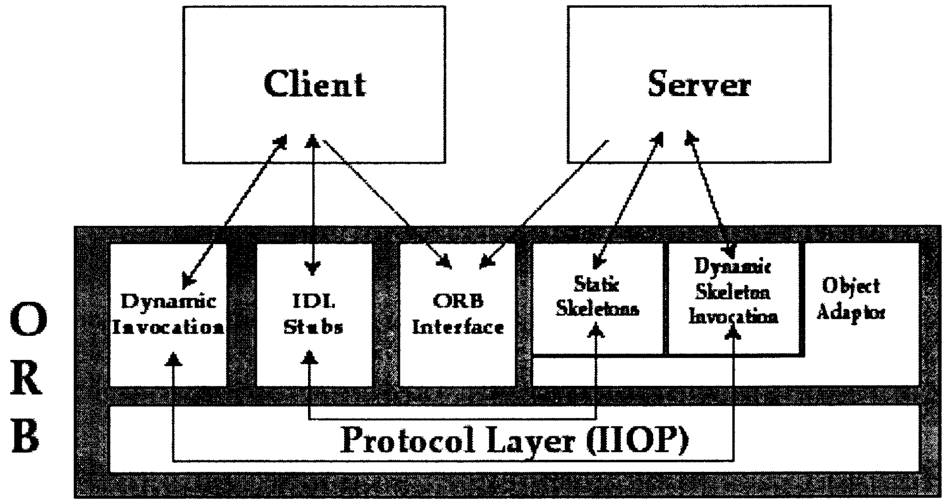

Figure 3.1 CORBA architecture.

For system integration, object implementation can be used to define interfaces for communicating with the data sources. The Stanford Digital Library Project is aimed at resolving the issues of heterogeneity of information and services have implemented this approach. Based on CORBA technology, the Information Bus (InfoBus) is the core system of the project that provides uniform access to heterogeneous information sources and services. This project, based at Stanford University, developed a modular Testbed infrastructure known as an information bus (or InfoBus) based on CORBA that enabled the integration of a variety of different digital library functions (Paepke et al., 1996). The InfoBus provides plug-in integration for repositories, information processing services, and user interfaces. It is implemented with CORBA distributed object technology. Heterogeneous repositories are wrapped with Library Service Proxy (LSP) objects that shield client programs from as much heterogeneity as technically feasible and appropriate from an end-user perspective. Library services (LS) built into the InfoBus provide the necessary support functions, such as query translation, metadata facilities, and rights management. Documents are modeled as objects. Their instance variables contain document fields, such as author or title. They are materialized from the underlying collections, which may or may not be object-oriented(Paepcke et al., 1998).The architecture of InfoBus is shown in Figure 3.2.

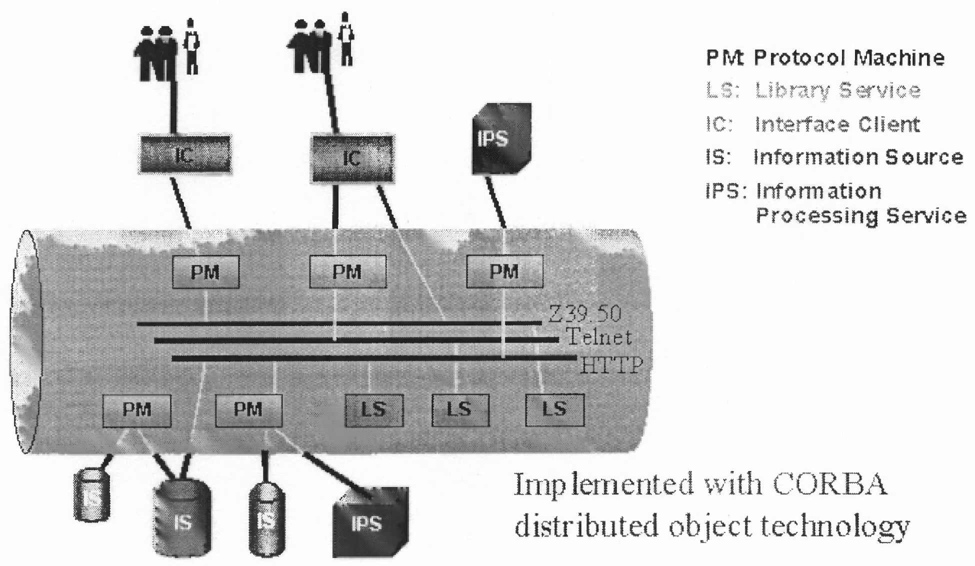

Figure 3.2 The InfoBus architecture of the Stanford Digital Library Project.

The question of compliance is very important in ORB. Even though ORB approach provides abstraction of the implementation of services at the object implementation, the task of data integration from multiple objects is done by the client. In order to do this object implementation the client application should know the metadata of the responses. ORBs developed by different vendors may have significantly different features and capabilities. Thus, developers of the client applications must learn a specification, the way vendors implement the specification, and their features. Also changes to the services provided by the data source require changes to the object application and propagation of the updated stub and skeleton. The mediated approach, discussed next, attempts to overcome these issues and limitations.

3.2 Mediated Approach

The mediated approach utilizes a component called mediator to perform integration. In this approach, the client sends the request. The integration system accepts the request, determine which set of information sources is capable to answer the request and generate the appropriate query plans for each information source. On obtaining the results from the information sources, the data integration system performs the appropriate translation, filtering and merging of the information and returns the final answer to the user or application (Fahmi et al., 2001). This process is referred to as a mediated approach, since the part that decomposes queries and combines results is often called the mediator.

The general architecture of mediated approach has the information sources, the wrappers, the mediators, and the user interface as shown in Figure 3.3. A wrapper is conceptually similar to a mediator, except that it only needs to translate between one native component and one mediator.

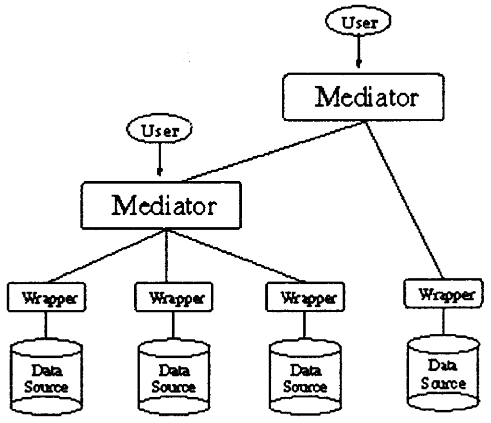

Figure 3.3 Mediated approach architecture.

The function of the wrapper is to interact with its corresponding information source, converting mediator queries represented in the common language into queries native to the sources and vice versa. To perform its task, a wrapper must have the knowledge of the underlying source. A wrapper connects the data sources with the mediator. The complexity of the wrappers depends on the amount of cooperation from the source itself. For example, a source can be cooperative by performing many of the processing tasks of answering the wrapper query. At the other extreme, a source may be non-cooperative, in which case, upon receiving an answer to a query from the source, the wrapper has to perform additional processing before sending it to the mediator. The function of the mediator is to accept incoming requests and transform them into the appropriate format. Each request is subdivided into smaller requests and sent to the appropriate source through the wrapper. Upon receiving answers to sub-queries, the mediator combines and integrates these answers to form the complete answer and presents it to the users. To perform its task, the mediator must have the knowledge of the sources and their schema to determine which sources provide what information.

The Digital Library Service Integration (DLSI) infrastructure at NJIT provides a systematic approach for integrating digital library collections and services. Users can see a totally integrated environment by using the DLSI. They can use their digital library system just as before. But in addition, they will see additional link anchors, and when they click on one, they will be presented with a list of relevant links. The architecture of the system is composed of Integration Manger which acts as the mediator, wrappers or enginelet, user interface and repositories. Figure 3.4 shows the architecture of DLSI. We will discuss this architecture in detail in the next section.

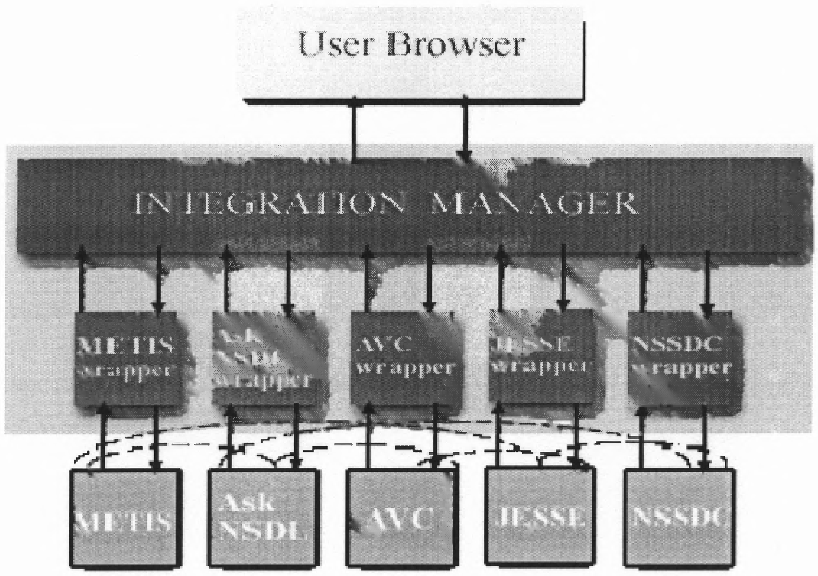

Figure 3.4 Architecture of digital library service integration at NJIT.

One drawback with the mediated approach is that the mediator component does not have the capability to search for new sources or discover potential sources that should be included in the integration. For this reason, agent technology has been introduced to overcome this limitation.

3.3 Agent-Based Approach

The approach is based on the idea of a software agent. An agent represents an element of the digital library (collection or service), and has the following special properties (Birmingham, 1995).

- **Autonomy:** the agent represents both the capabilities (ability to compute something) and the preferences over how that capability is used. Thus, agents have the ability to reason about how they use their resources. In other words, an agent not has to fulfill every request for service, only those consistent with its preferences. A traditional computer program does not have this reasoning ability.

- **Negotiation:** since the agents are autonomous, they must negotiate with other agents to gain access to other resources or capabilities. The process of negotiation can be, but is not required to be, stateful and will often consist of a "conversation sequence", where multiple messages are exchanged according to some prescribed protocol, which itself can be negotiated.

An agent is a component having decision making capabilities for performing different tasks. Agents can interact with the end users, with other related agents, and with the information sources. There are in general there types of agents which are used in the Agent-based approach. User agents, Provider agents (Back-end) and brokers (mediators). Architecture of agent-based approach is shown in Figure 3.5.

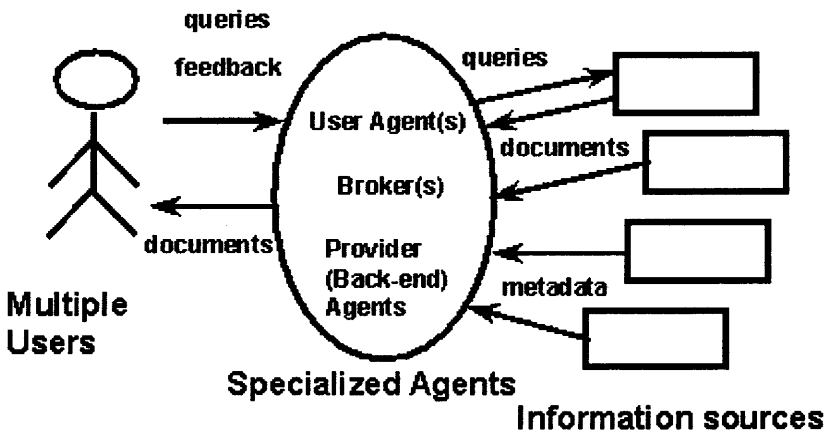

Figure 3.5 Agent-based approach architecture.

Brokers or mediator agents interact with user agents, provider agents and other brokers. There are many types of mediator agents in which each type performs specific intermediate tasks, including accepting user queries, evaluating user profiles if any, locating the appropriate source agents based on user queries, sending queries to appropriate source agents, monitoring the query progress, formatting and integrating responses from source agents, and communicating and working together with other mediator agents to accomplish a task (Wiederhold, 1992).

The brokers get the requests from the user agents and then locate the appropriate provider agents and pass the request to them. Brokers communicate with other brokers, monitor the progress, and integrate the responses from the providers to accomplish the request. To locate the appropriate agents, their description and services information is maintained in the repository. So when some service is needed, agents look in the repository. Also agents send their description to each other when ever requested. Upon receiving a response to the query, the original agent needs to update its knowledge base. This way, when it submits the same type of query for processing the next time, it can direct the query to the appropriate agent (Fahmi et al., 2001).

User agents get the requests from the users. They forward this request to the brokers along with the user profiles so that brokers can locate the user preferences. User agents also change the requests into proper format which is being used with in the system. Provider agents work just like wrappers as mentioned in the mediated approach.

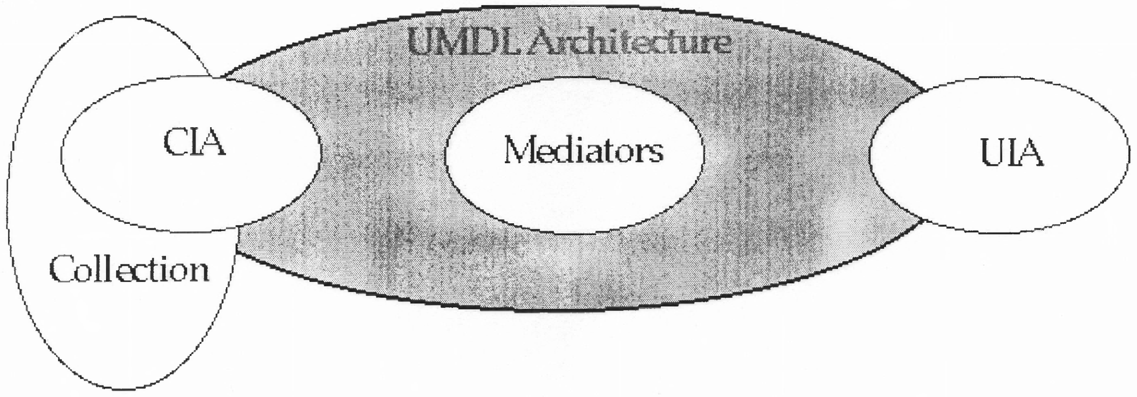

Figure 3.6 UMDL Architecture.

The UMDL is populated by three classes of agents (Birmingham, 1995).

- UIAs (User Interface Agents) provide a communication wrapper around a user interface. This wrapper performs two functions. First, it encapsulates user queries in the proper form for the UMDL protocols. Second, it publishes a profile of the user to appropriate agents, which is used by mediator agents to guide the search process.

- Mediator agents, of which there are many types, perform a variety of functions: essentially, all tasks that are required to refer a query from a UIA to a collection, monitor the progress of the query, transmit the results of a query, and perform all manner of translation and bookkeeping. Presently, two types of mediators populate the UMDL. Registry agents capture the address and contents of each collection. Query-planning agents receive queries and route them to collections, possibly consulting other sources of information to establish the route. Another special class of mediators currently being developed, called facilitators, mediate negotiation among agents.

- CIAs (Collection Interface Agents) provide a communication wrapper for a collection of information. While performing translation tasks similar to those performed by the UIA for a user interface, the CIA also publishes the contents and capabilities of a collection in the conspectus language.

3.4 A Service-Oriented Agent Architecture

We propose to implement the agent based approach using web services. A Web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided. To illustrate this distinction, one might implement a particular Web service using one agent one day, and a different agent the next day with the same functionality. Although the agent may have changed, the Web service remains the same.

A web service is Web-based application that can dynamically interact with other Web applications using an XML message protocol such as SOAP, XML-RPC or XMLP. Examples of emerging standards for describing, promoting and discovering these services are ebXML, UDDI and WSDL, and Microsoft's .NET and Sun's Sun ONE are major implementations of the concept. The goal is to enable one application to find another on the Internet that provides a needed service and to seamlessly exchange data with it.

Web services are utilized when you cannot or would not implement the logic yourself. A web service is defined by its messages, which must be self-sufficient referencing information necessary to understand the message. When using HTTP, the service reply is interpreted as responsive to the request sent on the same connection. XML is used for the self-describing interfaces and messages of web services.

SOAP is used to define message structure and supports the creation of complex self-contained messages. WSDL allows a service to define its interface that is basically the messages it will accept. UDDI supports design-time and run-time discovery of web services. The document-centric approach abstracts away the system architectures, creating loosely coupled connectedness that withstands changes to the underlying implementations (Burner, 2003).

A web service is an encapsulated chunk of behavior that is a self-contained and modular, self-describing using XML standards, programmatically and dynamically accessible over networks using standardized mechanisms such as SOAP, and capable of being dynamically composed with other Web services. By making its legacy data available via XML-based Web services, a business can greatly extends its reach to customers. The network economy is driving the evolution of e-business from rigid to flexible application design, from static to dynamic interaction between partners and from technology integration to business integration (Smith, 2001).

Although web services could be used in the agent-based approach to serve the purpose of integration, but the developer of the client applications has to use it manually, first searching the service and then finding the communication protocols. Developers of these services design registries like UDDI to be searched by the developers of the client systems. In order to overcome this manual process, a new approach has been proposed in this thesis to accomplish the task of integration in digital libraries.

CHAPTER 4

DIGITAL LIBRARY SERVICE INTEGRATION AT NJIT

Digital Library Service Integration (DLSI) project at New Jersey Institute of Technology is aimed at forming the core of vibrant virtual educational communities by supporting a broad range of community support services, beyond what most digital library researchers currently are developing. The DLSI infrastructure will provide the first step towards this vision. The middleware of DLSI is based on the mediated approach as described in this section.

4.1 DLSI Overview

The purpose of the Digital Library Service Integration project (DLSI) is to automatically generate links for digital library collections to related collections and services. Collections are libraries of computerized documents, which can include photographs, teaching modules, in addition to traditional types of documents. Services include searching, providing annotations and peer review. DLSI supplements collections by linking them automatically to relevant services and related collections. DLSI supplements services by automatically giving relevant objects in collections (and other services) direct access to these services. Users see a totally integrated environment, using their digital library system just as before. However, they will see additional link anchors, and when clicking on one, DLSI will present a list of supplemental links. DLSI will filter and rank order this set of generated links to user preferences and tasks". The DLSI infrastructure provides a systematic approach for integrating digital library collections and services. Digital libraries will be able to share relevant services within a seamless, integrated interface. Services and collections generally require minimal or no changes to plug into the DLSI infrastructure.

4.2 DLSI Architecture

DLSI architecture is based on the mediated approach. The DLSI infrastructure consists of four levels: an independent user interface, the Integration Manager, collection and service wrappers, and independent collections and services. The DLSI architecture is shown in the Figure 4.1.

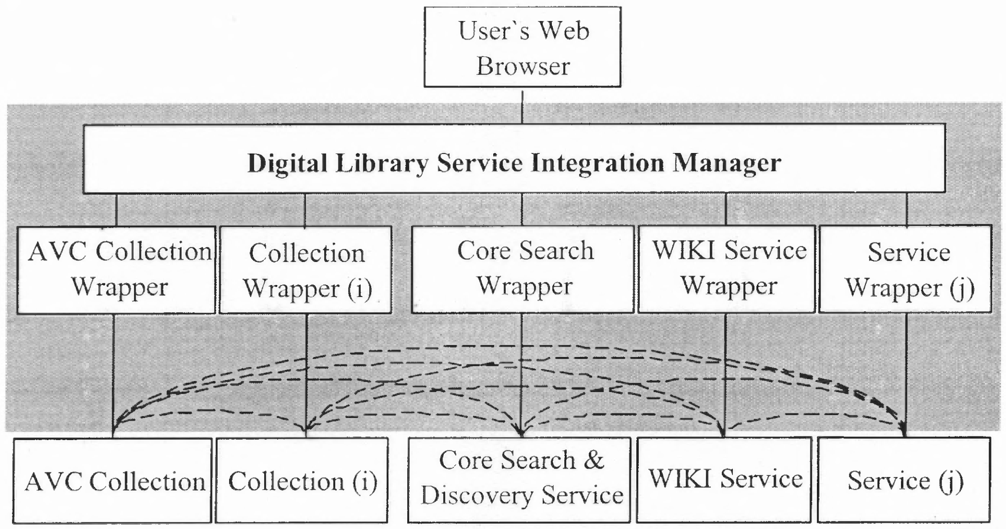

Figure 4.1 DLSI architecture showing wrappers, collections and integration manager.

4.2.1 Wrappers

A wrapper or enginelet must be developed for a collection or service to plug into the DLSI infrastructure. The wrapper's main task is to parse the display screens that appear on the user's Web browser to identify the "elements of interest" that DLSI will make into link anchors. First, wrappers will parse the display based on an understanding of the structure of its content. Second, DLSI will parse the display content using lexical analysis to identify additional elements of interest. If a service can operate on an element, DLSI will generate a link anchor over the element. Among the links generated for that anchor will be a link leading directly to that service's feature. Relationship rules specify which links DLSI will generate when the user clicks on a link anchor.

The NSSDC enginelet is the wrapper for National Space Science Data Center online system. This wrapper communicate with the integration engine which serves as the mediator as discussed in the mediated approach. The NSSDC enginelet is responsible for:

1. Intercepting the users search request to ensure that the results pass through the Metainformation Engine(ME).
2. Parsing the result screen and marking up elements of interest.
3. Dynamically generating links and commands associated with those links for all elements of interest.

The wrapper developer must focus on providing the parsing routines and mapping rules. These will not change as long as the application does not change. The routines and rules will apply to all instances of the user's screen (Bhaumik, 2003).

4.2.2 Collection and Services

Services and collections generally will require minimal or no changes to plug into the DLSI infrastructure. To integrate a collection or service with DLSI, an analyst must write a wrapper, initiate communications between the collection or service and the wrapper, and define relationship rules. The DLSI Integration Manager module manages relationship rules. Note that collections and services will continue to function as before. DLSI will supplement them for users who access them through DLSI's framework. All other users will use the services directly as before, and not see any of DLSI's supplemental features.

4.2.3 Integration Manager

In the DLSI architecture the Integration manager and wrappers together constitutes the Metainformation Engine (ME). The Metainformation Engine (ME) is a loosely coupled system, where various engine components communicate with each other via messages that conform to a well-defined standardized internal protocol. This approach allows new engine components to be developed and added without affecting existing engine components and functionality. The ME's goal is to supplement the output of most computer applications with link anchors and lists of links for each anchor, all with minimal or no changes to them (Bhaumik, 2003).

The ME v 1.0 has three primary components:

- The Engine Desktop translates the ME's internal messages, from the standard internal XML format to a format that can be displayed to a user via a web browser and vice versa.
- The ME Broker facilitates the communication between the Engine modules and works as the router for all the internal messages.
- The Mapping Rules Engine maps the data and relationships to hyperlinks at run-time. It maps the element instances in the application's output to the global element types (classes), and finds the links for them. Once the links are produced they are sent through to the engine desktop to be displayed in an appropriate interface to the user.

Message flow through the Metainformation engine illustrated below in Figure 4.2.

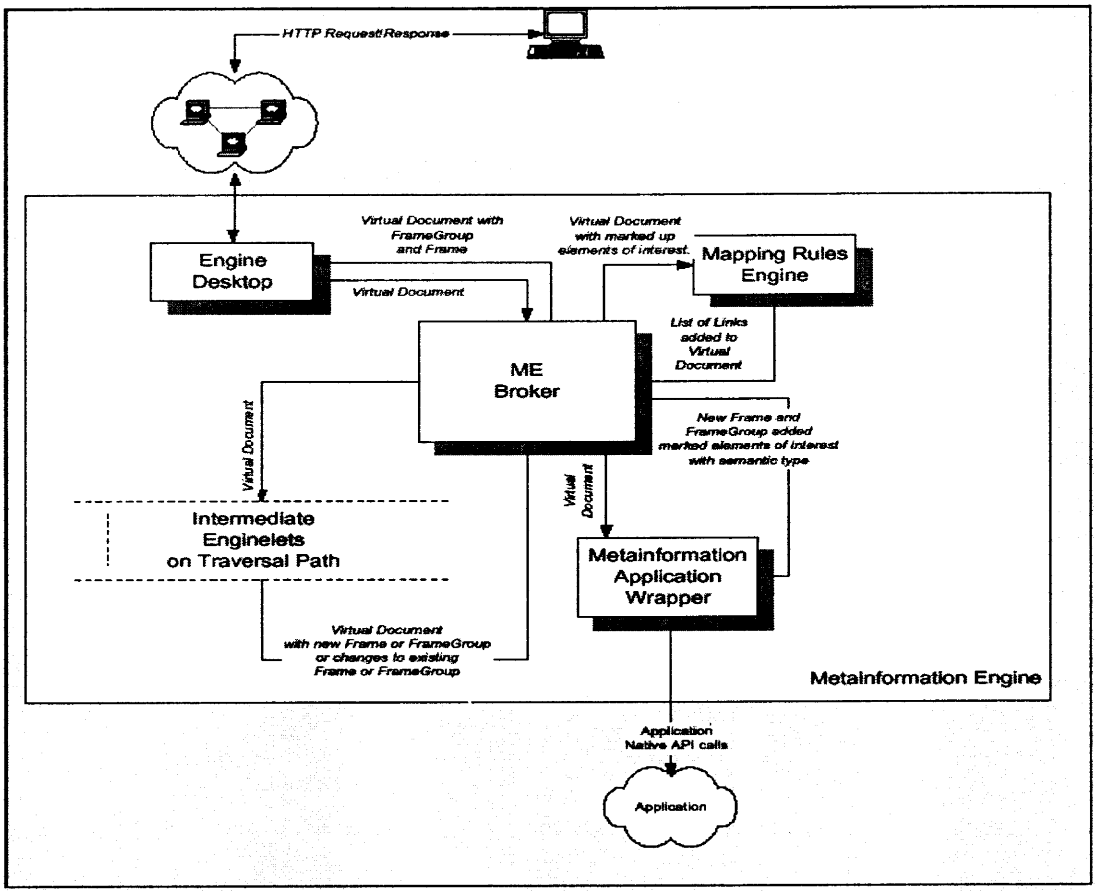
HTTP Request/Response
Engine Desktop
Virtual Document with FrameGroup and Frame
Virtual Document
Virtual Document with marked up elements of interest.
Mapping Rules Engine
List of Links added to Virtual Document
ME Broker
New Frame and FrameGroup added marked elements of interest with semantic type
Virtual Document
Virtual Document
Intermediate Enginelets on Traversal Path
Metainformation Application Wrapper
Virtual Document with new Frame or FrameGroup or changes to existing Frame or FrameGroup
Metainformation Engine
Application Native API calls
Application

Figure 4.2 Message flow in the ME

The following events occur in generating an interface to display to the user (Bhaumik, 2003).

- When a user follows a link to execute an engine command, the user's browser issues an HTTP request.
- The Engine Desktop generates a Virtual Document modeling the user's request. Virtual Document is the communication protocol among the various engine components
- The MEB(Metainformation Engine Broker) reads the source of the document (i.e., Engine Desktop) and the destination (the enginelet that will execute the command) and looks up the traversal path for that message through the Traversal Path Manger (TPM). MEB manages the communication between different components. The document is then routed through all intermediate enginelets in the order that they are listed in the traversal path. TPM

determines the intermediate enginelets that must process the document before reaching to final enginelet.

- The intermediate enginelets create/edit new Frames (and/or FrameGroups) or process the Virtual Document in some manner.
- After all enginelets in the traversal path have processed the message the message is sent to the destination enginelet - the MAW(Metainformation Application Wrappers). MAW is the component which communicates with the underlying application to identify the elements of interest. The MAW executes the command by communicating with the application via its native API. It then marks up the elements of interest in the application's output and adds the output document to a Frame and FrameGroup.
- This Virtual Document is then sent back to the MEB. The MEB again looks up the traversal path for this document, using the MAW as the source and the Desktop as the destination. The Virtual Document is again processed by all intermediate enginelets.
- After all enginelets in the traversal path have processed the message, the MEB passes the message on to the Mapping Rules Engine. The document contains the marked up "elements of interest" that the MAW located.
- The MEMRE(ME Mapping Rules Engine) looks up the mapping rules for "each element of interest" and generates the list of links. These list of links are added as Relationship objects in the Metainformation nodes of the Frames being manipulated.
- The Virtual Document is returned to the MEB, which then returns it back to its source desktop.
- The desktop uses the specified lens to translate each Frame and display embedded inside a FrameGroup for the user.

4.3 Implementation Details

XML (eXtensible Markup Language) and XSL (eXtensible Style Sheet Language)is a powerful technology that is currently being used to develop the system. XML/XSL is the functional backbone of the system's internal structure. The objects passed, as well as those that are regenerated, are delivered in the XML format.

JAVA is used in developing Integration Manager. It is used to process the XML objects that are passed between system modules. Using the JAVA platform, the XML documents are parsed for critical information. The current JAVA DLSI classes are used to gather the set of relational mapping rules that will be required to generate the newly implemented information links. JAVA is then again used to regenerate or create the resulting XML documents, which are then passed onto an XSLT document for transformation.

4.4 Advantages of DLSI

Collections and services will gain several benefits from integrating with DLSI:

- Users will have direct access to related collections and services, which in effect, enlarges the feature set of a given collection or service;
- Collections and services will gain much wider use, because DLSI linking will lead other users to them;
- Users will become aware of a service or collection from seeing its links included in DLSI's list of links when using other collections and services;
- DLSI will give the user direct, context-sensitive access to the features that a particular service or collection provides.

4.5 Disadvantages of DLSI

Disadvantage of the DLSI is that the component does not have the capability to search for new sources or discover potential sources that should be included in the integration. In order to integrate, developer must write a wrapper and the code to initiate the communication between the service and the wrapper. Also this wrapper or enginelet has to be registered with ME Broker.

To avoid all this manual processing we are proposing a new approach for the middleware to achieve seamless integration. This approach proposes the architecture to integrate the information on the web automatically (machine processable). Detailed architecture is explained in the next chapter.

CHAPTER 5

A NEW APPROACH TO DIGITAL LIBRARY INTEGRATION

This paper proposes a new approach to implement the current digital library service integration project to integrate NSSDC (National Space Science Data Center), AskNSDL and other repositories. It also introduces a new service to be used for the integration of a library database, ProQuest with any client application over the web. This approach will enable DLSI on the Semantic Web and will allow the use of metadata by the client applications and agents through the Semantic Web. The Semantic Web is a foresight for the future where information is given explicit meaning, which will help machines to automatically process and integrate information available on the Web.

There are various reasons that we propose semantic web services to be used as middleware to implement the integration in our new approach. By using the web services, DLSI will be available to client applications through a web service interface. Web services are lightweight components and by their use application developers can avoid the object models and programming language requirements. Moreover, ongoing Web services standardization efforts free them from the proprietary stigma of enterprise application integration systems (Vinoski, 2003). Web services are the evolutionary step in object-oriented programming for business-to-business and e-commerce applications. Tasks like real time auctions to get prices and performing multiple tasks like making traveling plan are not easily possible in Common Object Request Broker Architecture and Distributed Component Object Model and electronic data interchange (EDI) was too expensive and specialized to perform such kind of tasks.

Another reason for using this approach is that integration is the heart of web services (Fontana, 2001). Semantic web enabled web services provide most promising way to provide application to application integration. Ontologies are used to develop semantic web and are significant for applications that need to search and merge information from different resources. Ontologies provide automated reasoning ability which in turn is helpful in advanced services to intelligent applications such as semantic search and retrieval, software agents and intelligent databases. Ontologies are developed in a logic based language which helps to provide detailed, precise, steady, sound, and meaningful differences among the classes, properties, and relations. Ontologies will provide many benefits for the digital library integration project.

This new approach will allow more intelligent syndication. For example on the ProQuest website a simple list of subject areas may not provide the users sufficient ability to search for the contents they want. Ontologies are used to describe content and axioms that define terms. By using these definitions other facts which are necessarily true could be inferred. These inferences can allow users to obtain search results, which are impossible to obtain through conventional retrieval systems. But this approach depends on digital resource providers annotating their pages with the ontologies.

Ontologies could also be used to provide semantic definitions of multimedia collections. Multimedia ontologies can be media specific or content specific. Media specific ontologies provide taxonomies of the media types and to explain properties for example scene breaks and length of clips. Content specific ontologies describe the subject of media like participants. In the retrieval process this information will be used to get the semantics from the multimedia collection which is not possible in current integration project.

Another benefit of this approach is that the semantic web can provide agents with the capability to understand and integrate diverse information resources (Mikhalenko, 2003). Intelligent agents take instructions from users for example “make a complete traveling plan in January” and search the internet to produce optimum results. When completing the task the information may come from various sources such as service specific sites, reservation sites etc.

By using the web services, client applications will seamlessly interoperate with them without concerning about their location. Semantic web enable web services will make the integration process much easier and automated as these services will be machine searchable and processable. In this way this new approach will be able to overcome the short comings of other approaches. In ORB approach question of compliance is a big issue the client must know how the vendor has implemented the architecture. Also in mediated approach, the mediator component does not have the capability to search for the new sources. In the DLSI architecture every new wrapper must be developed and registered when client applications want to integrate through DLSI. Also the generated list of links is very limited and non inferable. In order to add links for more digital resources, wrappers must be modified.

With this new approach all the integration will be machine processable and no extra coding will be required for new applications to be integrated. Once the digital resource pages will be semantic web enabled and web services will be developed, any application can integrate itself automatically. The most difficult and important part in this approach is to annotate the contents of digital resources into web ontology language. Once the content is converted any client application can access it using the proposed semantic web enabled web services. These web services will provide the mechanism to be used automatically by other machines. In regular web services the client application developers must know how to communicate with the services through their interface but with the semantic web enabled web services applications can directly communicate with them without the involvement of any human. In this way once the setup is complete, any application over the internet can use this mechanism to integrate itself with the digital resources. The only limitation in this approach is that the conversion of digital resources into web ontology language is very time consuming and slow process. But once the large amount of information will be converted into ontologies, this approach will provide the most effective and efficient way of digital integration.

Basic architecture of the new approach is similar to that of Agent-Based approach implemented through web services. Although web services serve the purpose of integration, but this is a manual process as registries like UDDI are designed to be searched by the developers of the client systems. In contrast, this new approach uses semantic relations to find services described using Semantic Web languages. This approach is conceptually based on the Semantic Web and Ontologies. Semantic web concept is still in its early stages but soon it will dominate the current technology.

This thesis proposes the architecture for two integration projects. DLSI will be implemented as semantic web services using ontology languages. Ontologies are significant for applications that need to search or merge information from different resources. In case of DLSI web service, the repositories like NSSDC and AskNSDL will communicate with the digital resources through this service. Once these repositories will pass the pages to DLSI web service, links will be created which connects to the available digital resources. In first phase the NJIT library contents will be written in the web ontology language like DAML-S. So for example whenever a user will be looking for extra information from NSSDC page, ontologies will be employed to infer and get the related information. The ProQuest web service will enable the programmatical access to the full text document discovery from the collection. All the contents will be converted to web ontology language. So in result any web application can use this web service to integrate the information on their pages with the information in the ProQuest digital resources. Detailed proposed architecture has been discussed latter in this chapter.

5.1 Semantic Web Overview

5.1.1 Semantic Web

The Semantic Web is a network of information linked up in such a way as that it could be easily accessible globally. This information is machine useable and associated with more meaning. The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation *(*Berners-Lee et al., 2001*)*. The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. Semantic web could be seen as a huge engineering solution. Semantic Web vision of a machine-readable web has possibilities for application in most web technology.

Today's web was designed for human use. The current Web supports documents, pages of text and figures designed for humans. But there is an increased automation in terms of direct operation between machines, mostly in B2B, B2C and information services applications. Currently this is done by APIs which needs hand coded information retrieval code on the client side. Fundamental to having computer programs or agents implement reliable, large-scale interoperation of Web services is the need to make such services computer interpretable, to create a Semantic Web of services whose properties, capabilities, interfaces, and effects are encoded in an unambiguous, machine-understandable form (McIlraith et al., 2001).

The Semantic web adds support for databases, vast collections of information organized to be processed by machines. Machines can parse HTML but cannot reliably infer any semantic information from this attempt. The Semantic Web aims at bringing meaningful content out of Web pages. Machine usable content means that the machine knows what to do with information on the Web.

5.1.2 Resource Description Framework

W3C has hailed RDF (Resource Description Framework) as a Semantic Web language to implement the concept of Semantic web. The information representation on the semantic web is done by Extensible Markup Language (XML) and the Resource Description Framework (RDF). XML allows users to add structures to their documents, but the developer should know these structures, in order for programs to use these. XML does not capture information about what the structures mean. On the other hand RDF enables users to use Metadata to describe data on the Web. RDF gives you a way to make statements that are machine-processable. Machines can infer relationships between resources on the basis of metadata. The Semantic Web will develop on XML's ability to define customized schemes and RDF's flexible approach to represent data.

5.1.3 Uniform Resource Identifier

Identifiers are used to identify items on the web. As we use uniform system of identifiers, and because each item identified is considered a "resource," we call these identifiers "Uniform Resource Identifiers" or URIs. RDF statements are composed of URIs. All the information on the web and in the databases will have a URI and can be accessed through RDF.

5.1.4 Ontologies

The Semantic Web requires a language which can formally describe the semantics of classes/sub-classes and properties used in web documents. To perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema. Web Ontology language is used for this purpose.

In order for the semantic web to succeed, programs must be able to compare or combine information across different systems. A well-known problem with the web of today is finding the many web services currently available on line. Ontologies are used for this purpose. Ontologies are collection of information which describes common meanings and relationships between resources on the Web. Ontologies include machine usable definitions of basic concepts and the relationships among them. One of the most powerful uses of the web Ontologies is in the area of web services. The Semantic Web needs ontologies with a structure to provide description about things in the different domains, relationships among them and attributes of those things. Ontologies are very important for digital library applications that want to search across or merge information from diverse communities.

The standard W3C ontology language is OWL, which stands for Web Ontology Language. The RdfSchema is one such language which provides a means for formalizing ontologies. RdfSchema extends the Resource Description Framework model by enabling a collection of resources to be described according to a simple class hierarchy. However, RdfSchema is a simple ontology language and in order to achieve interoperation between autonomously developed and managed schemas, richer semantic language is required. A DAML Ontology (DAML-O) is a document that describes a vocabulary of terms for communication between human and automated agents.

Ontologies can be used in an integration task to describe the semantics of the information sources and to make the content explicit. With respect to the integration of data sources, they can be used for the identification and association of semantically corresponding information concepts (Wache et al., 2001).

5.1.5 Ontologies and Services

Web ontologies most useful application is in the area of web services. Semantic Web Services are used to implement the concept of semantic web. In many B2B and E-Commerce applications the web services utilize APIs to locate and extract content from the pages. But if the page changes then the web service must be rewritten. In addition, all programs that wish to utilize it must understand the interface description of web services. Also problem with the web of today is the difficulty in finding the many web services currently available. What is needed is semantic web service whose properties, capabilities, interfaces, and effects are encoded in an unambiguous machine-readable form (McHraith et al, 2001).

The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications. The Web can reach its full potential if it becomes a place where data can be shared and processed by automated tools as well as by people (Miller, 2004).

Any world wide web application could be converted to web service by providing an Application Programming Interface (API) to that service, an access protocol such as SOAP and a layer describing the service such as WSDL. The transition from a Web-Service to a Semantic Web Service requires expressing the processes that comprise that Web-Service using the Ontology Web Language for Services and any supporting ontology encoded using the Ontology Web Language (Bechhofer et al.,2004).

To implement Semantic Web services, a markup language must be descriptive enough that a computer can automatically determine its meaning. That language must perform the following tasks.

- **Discovery:** A software must first be able to automatically discover, an appropriate Web service for the task in hand. A Semantic Web service describes its properties and capabilities so that software can automatically determine its purpose.
- **Invocation:** Software must be able automatically to determine how to invoke or execute the service. A Semantic Web service provides a descriptive list of what a requester needs to do to be able to execute and fulfill the service. This includes defining the inputs and outputs of the service.

- **Composition:** Software must be able to select and combine a number of Web services to complete the task in hand. The services have to interoperate with each other seamlessly so that the integrated results provide the solution.

Ontologies should be used to create the services in any ontological language. Client applications can use the hierarchy to find matches via the class/subclass properties or other semantic links. For example, someone looking for any specific information about the planet Mars might find NSSDC (collects information about planets) even if there were no exact match for that specific information about Mars. This is possible due to the expressiveness of web ontology languages like DAML to develop semantic web enabled web services. DAML is used to describe the structure in terms of classes and properties and this structure is then used for inference and reasoning. Also by using description logic and other inferential ways user can find information that was not explicit. We can also include machine readable description of a service and some explicit description of the consequences of using the service. In this way we can integrate ontologies and agents.

5.1.6 Ontologies and Agents

The real power of the Semantic Web will be realized when independent computer programs called agents are created that collect and reason over Web content from varied sources, exchanging data and working cooperatively with other agents.

The exchange of proofs is an important aspect of agents functioning, and agents will be able to convert their internal reasoning into a common representation language (such as RdfSchema) so that other agents can check their reasoning. Ontologies allow explicit organization of knowledge in agent-based applications, and unambiguous description of characteristics and properties of agents (Zini et al., 1999).

5.2 Ontologies Based Approach

Standard Web Service technology provides limited support in automated service recognition and combination, service comparison, and automated negotiation. In digital libraries there is a need for automatic cooperation between available services. Any system interaction with another application needs automatic discovery and selection of the optimal Web Services.

This thesis proposes a new architecture for the middleware implementation. This new architecture, named Ontologies Based Approach, is primarily based on Web Services and Semantic Web. By using web services the service interface will be published on the web using XML and will be invoked by the client applications using HTTP and SOAP. Through semantic web will provide the web published and accessable semantic description through dynamically linked and shared ontologies.

Interoperability is a key application of Ontologies and many ontology based approaches has been developed for information integration in order to achieve interoperability (Uschold et al., 1996). In order to design a self-regulating Web Service Ontology, it is necessary to understand how it will automatically work. Designing a Semantic Web Service requires detailed analysis that exposes and eliminates semantic logic conflicts and indeterminacies.

5.2.1 Markup Language for Ontologies

Management of resources in Semantic Web is impossible without use of ontologies, which can be considered as high-level metadata about semantics of Web resources (Fensel et al., 2002). DAML-S is an upper ontology markup language for describing properties and capabilities of Web Services. DAML-S provides an unambiguous, computer interpretable markup language, which enables automation of service use by agents and reasoning about service properties and capabilities (Ankolenkar et al., 2001).

The basic requirements to service description language in (Trastour et al., 2001), formulated as:

- High degree of flexibility and expressiveness;
- Ability to express semi-structured data
- Support for types and categorization;
- Ability to express constraints.

Considering these requirements and comparing proposed by Semantic Web ontological descriptions (e.g. DAML-S) with other layers as shown in Figure 5.1.

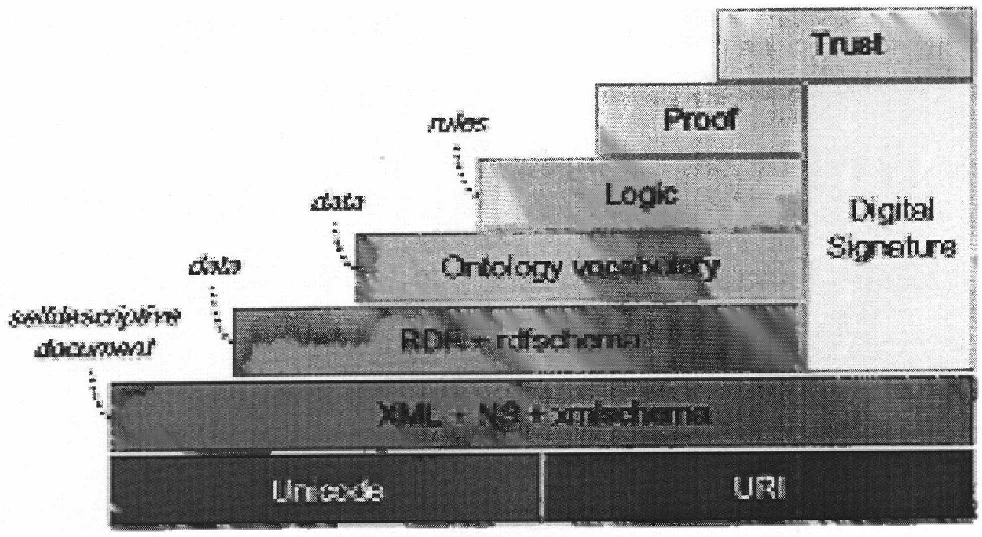

Figure 5.1 Layered approach to markup language development.

In DAML-S: RDF layer provides the flexibility, expressiveness and semi-structure data, RdfSchema layer provides the support for the types, whereas DAML layer meets the constraint requirement.

5.2.2 Digital Library Web Services and Architecture

We will use DAML-S to develop the web services. Ontologies will be used to facilitate information retrieval services to the clients. When users will follow a hyperlink generated by our web service or search for some information, these ontologies will be used to direct the users to related information providers domains over the internet. When the client applications look for some information through agent brokers, they will use the inference and reasoning capabilities of ontologies and direct to related digital documents. Agent brokers are used to carry out requests in an open market way from the distributed collection of digital documents.

The DAML-S ontology is conceptually divided into three subcategories. Profile ontology describes what a service does and provides the way for its discovery. Process specifies the working of the service including the internal process and dataflow. Grounding provides the information about the implementation of the service. Accordingly, each DAML-S description has three major parts.

The two specific Semantic based web services that are being proposed are as follows.

1. **DLSI web service:** This web service will provide all the services that are currently available in the DLSI architecture. Web service will produce hyperlinks in the web pages for related information. Requesting application will provide the document to the web service. DLSI web service will process the document and will generate the augmented links and send it back to the requesting application. DLSI web service will be implemented using DAML-S. In the profile part of the ontology, information about the service provider, inputs(documents types), outputs(augmented documents) and other preconditions will be explained. The process part of the ontology will explain the process model and internal process and dataflows. The documents processing implementation(parsing and augmentation) will be in this part of the ontology. Finally the grounding will explain the operation level details. It will provide the details about the message exchange formats and network protocol and hence will enable the automatic invocation of the service by

other machinces. In this way this web service will be machine processable. In the first phase NJIT library digital resources will be converted to semantic web enabled and hyperlinks will look for the related information in these resources. Latter on more digital resources will be added for this service. Ontologies will allow reasoning for the search of related information.

2. **ProQuest web service:** This paper also proposes a web service to allow the integration of ProQuest services in the semantic web. This web service will allow other web based applications to programmitacally access the ProQuest services in a same way as human user can access through ProQuest web interface. ProQuest web resources will be annotated as web ontology language which will make it semantic web enabled. This can be taken advantage by the brokers on the semantic web. ProQuest webservice provides the facility to lookup the articles from the ProQuest data library and show it in an integrated way. The requesting applications will envoke the service by sending a string of text and in return will get the related articles. The string could be sent by following the hyperlinks or by quering the specific text from the client applications. ProQuest webservice will get the string and in return will provide the construct. ProQuest webservice will be developed using DAML-S. The layered description will be used including profile, process and grounding. Requesting applications will inspect the profile to see if the service has the desired capability and how to interact. Process Model will provide detailed information about how the service works. If the requesting application sees that the service conforms to the requirements, the grounding will specify the implementation details needed for executing the service. The service will be automatically invoked by applications or agents.

Similar to agent based approach, in ontologies based approach different tasks will be accomplished by Semantic Web agents (service providers, requesters, and middle agents). Our architecture will assume the following capabilities of agents

- Semantic web agents should be able to interpret published ontologies and can send and understand messages with content represented as published ontologies.

- Requester agents, could be web services or clients, must be able to send requests to servers in terms of their published and semantic interfaces.

- Service provider agents must publish their semantic descriptions about the interaction procedure and capabilities. Through this information requester agents will interact with these services. Ontologies will be used for these descriptions.

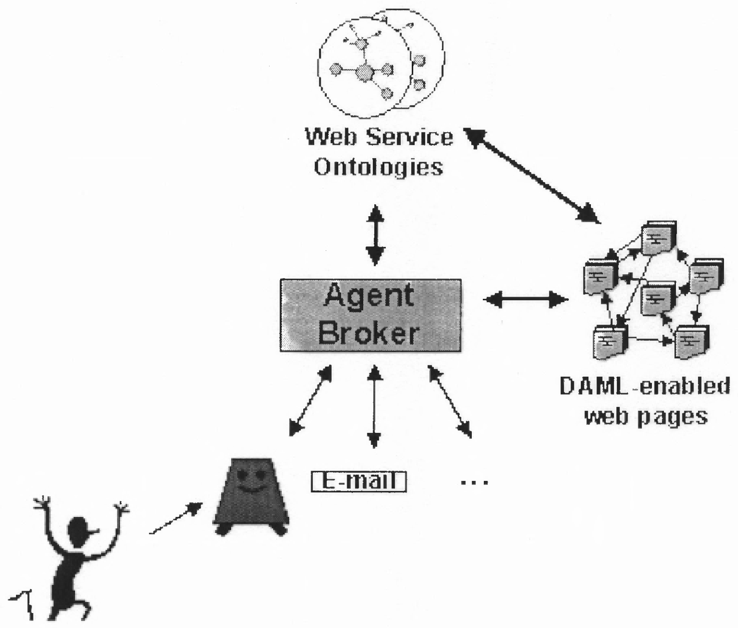

Figure 5.2 DAML-enabled agents, tasked by users or other agents.

DLSI web service and ProQuest web service as discussed earlier will serve as service provider agents and will be used for three purposes.

- To describe collection content and inference rules to support powerful user queries. In case of DLSI web service these queries means to find the related contents in the digital resources and applying inference rules. In ProQuest web service the articles will be searched using inference rules from the ProQuest database.
- To enable agent brokers to use the service.
- For the Copy right material, licensing mechanism will be implemented.

The two web services must allow the three basic tasks of automatic discovery, automatic composition and automatic invocation to make the services semantic web enabled. The functional requirements of these tasks will be met using ontologies. The architecture is shown in the Figure 5.3.

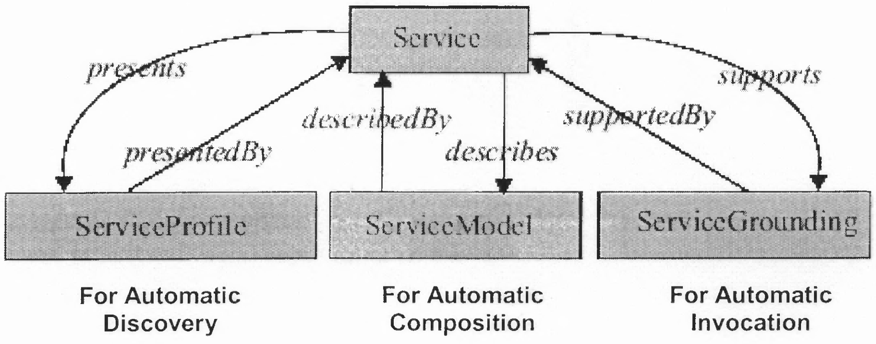

Figure 5.3 NJIT digital library web service functional requirements.

1. **Automatic Discovery:** Web services present the service profile. Service profile contains the service provider information, and also specifies the inputs, outputs and preconditions of the service. This information is used by the client applications and agents to use the service automatically, without manually finding the service. Automatic discovery is the process by which a client, interacting with other clients or middle agents, identifies candidate services to achieve the client's objectives. Any web service must provide this mechanism in order to become semantic web enabled. The DLSI web service and ProQuest web service profiles will provide all the necessary information and could be discovered automatically by the client applications who wants to integrate their applications using these services.

2. **Automatic Composition:** Service model is described by the services and is used for the automatic composition or execution of the web service. It describes how the subsystems in the service work. The task is accomplished by the automatic selection, composition, and interoperation of Web services to perform some complex task provided a high level description of an objective. Automatic composition comes into picture when multiple services are utilized to achieve an objective. Suppose someone wants to make all the travel arrangements to a conference. Conference registration, airline ticket, hotel reservation and other related activities will be carried automatically rather than going to the individual websites.

3. **Automatic Invocation:** Service grounding will also be supported by the services. It is used for automatic invocation or interoperation. Automatic invocation is used by the computer program or agents, given only a declarative description of that service. This is used in the semantic web enabled web services as opposed to the regular web services where agents are pre-programmed to be able to call that particular service. The agent will be able to understand what input is necessary to invoke the service and what will be the output from the service. For example, a user can request the purchase of an article from a specific journal using ProQuest web service, from any other website located by searching and then selected by that user.

5.3 Implementation Details

Web Services Description Language (WSDL) provides an encoding and protocol independent mechanism to describe the means of interacting with offered services. WSDL is an XML-formatted language used to describe a Web service's capabilities as collections of communication endpoints capable of exchanging messages. WSDL is an integral part of UDDI, an XML-based worldwide business registry. WSDL is the language that UDDI uses. WSDL separates the abstract definition of service and messages from their concrete binding to a network port and message format. The current WSDL specification describes concrete bindings for SOAP, HTTP, and MIME.

In DAML-S Service Profile is used for the description of a Web service which tells the service locators about what the service does. A Service Model tells about how the service works, and Grounding tells the requestors how to access the service. The Service Profile and Service Model are abstract specifications as they do not specify the details about message formats, protocols, and network addresses by which a Web service is instantiated. On the other hand the Service Grounding provides these more concrete details. The Web Services Description Language (WSDL) provides a well defined means of specify these kinds of details. Therefore, in the proposed ontology based approach, WSDL will be used to ground DAML-S services.

DAML-S is an XML-based language, and its process declarations and input/output types fit with WSDL. Therefore it is much easier to extend WSDL bindings to use with DAML-S, like the SOAP binding. For this new approach DAML-S specified arbitrary atomic process can be given a grounding using WSDL and SOAP, and HTTP could be used as transport mechanism. In order for Grounding DAML-S with WSDL and

SOAP the construction of a WSDL service description will be used with all the parts (message, operation, port type, constructs, and binding).

This new approach as discussed in this section will serve a middleware. World Wide Web users will access the web applications, these applications will communicate with the semantic web agents to accomplish the tasks. Semantic web enabled web services will serve as a middleware to do the seamless integration between the applications.

CHAPTER 6

CONCLUSION

The goal of this thesis is to assess the working of currently employed middleware used for digital library integration, and propose a new architecture which overcomes the limitations of earlier approaches. This new ontologies-based approach utilizes the semantic web-enabled web services to implement the middleware for digital library integration. Paper provides the details to convert the mediated based DLSI architecture at NJIT into ontologies based DLSI. It also envisions a new service to do the seamless integration of applications with ProQuest data repositories.

This new approach will make it easier for machines to automatically process and integrate information available on the web. In this way the new approach provides a more natural and flexible way of integrating the applications. The Semantic web is an opportunity for web services to reflect their organization and show their value-added. We believe that the approach proposed here for migrating the DLSI into semantic service could be used as a model for other integration projects.

The main challenge in using this approach is converting the pages to semantic web enabled. There are over one million pages over the web and are potential resources for any digital library. In order for successful and effective digital libraries more and more contents should be written in the web ontology languages so that the full benefit of semantic web could be achieved.

We encourage the further research to use these guidelines to develop and implement this system. This will enhance the capabilities and functionalities of currently used DLSI project and will be helpful in overcoming the shortcomings of the current middleware approach.

REFERENCES

Adam, N. R., Atluri, V., and Adiwijaya, I. (2000). SI in Digital Libraries. Communications of the ACM, 43(6), 64-72.

Ankolenkar, A., Burstein, M., Cao Son, T., Hobbs, J., Lassila, O., Martin, D., D., McDermott, McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., and Zeng , H.(n.d.): DAML-S: Semantic Markup For Web Services. Retrieved November 22, 2004, from DARPA agent markup language page. Web site : http://www.daml.org/services/daml-s/2001/10/daml-s.html.

Barrett, D.J., et al. (1996). A Framework for Event-Based Software Integration, ACM Transactions on Software Engineering and Methodology, 5(4), 378-421.

Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Schneider, P. and Stein, L. (2004) OWL Web Ontology Language Reference, Retrieved January 01, 2005.
Web site: http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

Berners-Lee, T., Hendler, J., Lassila O. (2001). The Semantic Web. Scientific American, 284(5), 34-43.

Bernstein, P.A.(1996), Middleware: A model for distributed systems services, Communications of the ACM, 39(2) , 86-98.

Bhaumik, A., Bieber, M.(2003). Metainformation Engine v1.2. Technical Report. Retrieved November 10, 2004, New Jersey Institute of Technology, IS Department. Web site: http://www.is.njit.edu/dlsi/docs/ME%20Architecture-V4-Final.pdf.

Birmingham, B., et al.(n.d.). EU-NSF Digital Library Working Group on Interoperability between Digital Libraries. Retrieved November 20, 2004, from National Research Council, Italy. Web site : http://www.iei.pi.cnr.it/DELOS/NSF/interop.htm.

Birmingham, P. W.(1995). An agent-based architecture for digital libraries. D-Lib, Retrieved November 20, 2004 from D-Lib Magazine. Web site: http://www.dlib.org/dlib/July95/07birmingham.html.

Bray, M. (n.d.), Lockheed-Martin Ground Systems. Retrieved November 10, 2004, from Carnegie Mellon University: Software Engineering Institute. Web site: http://www.sei.cmu.edu/str/descriptions/middleware_body.html.

Burner, M.(2003), The Deliberate Revolution. Creating Connectedness with XML Web Services. ACM Queue, 1(1), 29-37.

Emmerich, W.(2000), Software Engineering and Middleware: A Roadmap. Proceedings of the conference on the future of software engineering. 117-129.

Fahmi, I., and Ibrahim, I. K. (2001). Using The OAI Metadata Harvesting Protocol as a Simple and Effective Data Integration Framework for the Digital Library Network at the Third World. Retrieved October 29, 2004 from The Indonesian Digital Library Network. Web site: http://idln.itb.ac.id/Open.html?target=papers/maroko-ismail-oai.htm.

Fensel, D., Hendler, J., Lieberman, H., and Wahlster, W. (2002), Semantic Web Technology, MIT Press, Boston.

Fontana, J., (2001). Web Services: Where middleware and XML converge. Retrieved December 31, 2004 from Networkbuzz world.
Website: http://www.nwfusion.com/buzz2001/webserv/.

Greco, S., Pontjeri, L., and Zumpano, E. (2002). Lecture Notes In Informatics (Lni) Proceedings of the 2001 international conference on Information systems technology and its applications, 2, 75 – 84.

Hasselbring, W.(2000). Information System Integration. Communications of the ACM, 43(6).32-38.

Ibrahim, I. K.(2001). Semantic Query Transformation for the Intelligent Integration of Information Sources. Ph.D. thesis. Gadgah Mada University, Indonesia.

Linthicum, D. S. 2004. Next generation application integration : from simple information to Web services, Addison-Wesley.

McIlraith, S. A., Son, T. C., and Zenf, H. (2001). Semantic Web Services. IEEE Intelligent Systems. 16(2), 46-53.

Mikhalenko, P., (2003). The benefits of the Web ontology language in Web applications. Retrieved Januray 01, 2005. Web site: http://builder.com.com/5100-6387-5060266.html.

Miller, E., (2004). Semantic web activity statement. Retrieved January 01, 2005.
Web site: http://www.w3.org/2001/sw/Activity.

Nilsson, E.G., Nordhagen, E.K.; Oftedal, G.(1990). Aspects of Systems Integration, Proceedings of the First International Conference on Systems Integration, 434 – 443.

Paepcke, A., Cousins, S.B., Garca-Molina, H., Hassan, S.W., Ketchpel, S.P., Rscheisen, M., Winograd, T.(1996). Using distributed objects for digital library interoperability. IEEE Computer, 29 (5), 61-68.

Paepcke, Andreas, Baldonado, Michelle; Chang, Chen-Chuan K.; Cousins, Steve; Garcia-Molina, Hector.(1998) Building the InfoBus: A review of technical choices in the Stanford Digital Library Project , Retrieved January 02, 2005. http://dmn.netlab.uky.edu/~seales/cs585/case-study-papers/stanford-infobus.pdf.

Papazoglou, M. P., & Georgakopoulos, D. 2003. Service-oriented computing, Communications of the ACM, 46, 24-28.

Parent, C. and Spaccapietra, S. (1998). Database Integration: an Overview of Issues and Approaches. Communications of the ACM, 41(5), 166-178.

Sage, A.P.(1992). Systems Engineering., John Wiley & Sons, Inc., New York.

Sikora, R., Shaw, M.(1998). A Multi-Agent Framework for the Coordination and Integration of Information Systems, Management Science, 44(11), 65-78.

Smith, R.(2001), Trends in e-business technologies, IBM Systems Journal, 40(1), 4-7.

Trastour, D., Bartolini, C., Gonzalez, J.(2001), A Semantic Web Approach to Service Description for Matchmaking of Services, In: Proc. International Semantic Web Working Symposium (SWWS), Stanford, CA, USA, Track 3.

Uschold, M., Gruninger, M.(1996). Ontologies: Principles, methods and applications. Knowledge Engineering Review, 11(2),93-155.

Vinoski, S., (2003). Integration with Web Services. Retreived December 31, 2004. Website:http://www.iona.com/hyplan/vinoski/pdfs/IEEEIntegration_With_Web_Services.pdf.

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S.(2001). Ontology-based Integration of Information - A Survey of Existing Approaches. In Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, 108-117.

Wiederhold , G.(1992), Mediators in the architecture of future information systems. IEEE Computer, 25(3), 38 – 49.

Zini, F. and Sterling , L.(1999). Designing Ontologies for Agents. In M. C. Meo and M. Vilares-Ferro, editors, Proc. of Appia-Gulp-Prode'99: Joint Conference on Declarative Programming, L'Aquila, Italy, 29-42.