# A WEB SERVICES / AGENT BASED MODEL FOR INTER-ENTERPRISE COLLABORATION

By

Akbar Siami Namin

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
LAKEHEAD UNIVERSITY
THUNDER BAY, ONTARIO, P7B 5E1 CANADA
JANUARY 2004

*Your file Votre référence*
*ISBN: 0-612-96996-7*
*Our file Notre référence*
*ISBN: 0-612-96996-7*

*To My Parents*

# Table of Contents

List of Tables vii

List of Figures viii

Abbreviation ix

Abstract xi

Acknowledgements xii

1 Introduction 1
- 1.1 Global Competition and Enterprise Collaboration . . . . . . . . . . . 2
- 1.2 Scope of the Thesis . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
- 1.3 Organization of the Thesis . . . . . . . . . . . . . . . . . . . . . . . 8

2 Literature Review 10
- 2.1 Concepts Review . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
  - 2.1.1 Virtual Enterprise . . . . . . . . . . . . . . . . . . . . . . . . 10
  - 2.1.2 Web Services . . . . . . . . . . . . . . . . . . . . . . . . . . 14
  - 2.1.3 Software Agents . . . . . . . . . . . . . . . . . . . . . . . . 18
  - 2.1.4 Web-Services vs. Software Agents: A Comparison . . . . . . . 21
- 2.2 Literature Review . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
  - 2.2.1 Web Services/Agent-based Approaches for Workflows . . . . . 31
  - 2.2.2 Agent-based Approaches for Service Composition . . . . . . . 36
  - 2.2.3 Agent-Based Approaches for Quality of Web Services . . . . . 42
  - 2.2.4 Utilizing Software Agents in Web Services Technology . . . . . 46
  - 2.2.5 Examples / Related Projects . . . . . . . . . . . . . . . . . . 48
- 2.3 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52

**3 A Web Services / Agent Based Model for Inter-Enterprise Collaboration 53**

3.1 Enterprise Collaboration: Challenges . . . 53
3.2 Integration of Web Services and Software Agents . . . 56
3.3 A Web Services / Agent Based Model . . . 61
3.3.1 System Architecture . . . 62
3.3.2 Enterprise Model . . . 63
3.3.3 UDDI Servers Model . . . 78
3.4 Summary . . . 86

**4 Software Prototype System Design and Implementation 87**

4.1 System Design . . . 87
4.1.1 Challenges . . . 88
4.1.2 System Architecture . . . 88
4.2 System Interactions . . . 95
4.3 A Scenario . . . 98
4.4 Prototype Implementation . . . 101
4.4.1 Tools Used for Prototype Implementation . . . 103
4.4.2 Software Components . . . 105
4.5 Summary . . . 109

**5 Conclusion and Future Work 110**

5.1 Conclusion . . . 110
5.2 Future Work . . . 112

**Bibliography 116**

# List of Tables

2.1 Agents and Web services corresponding basic requirements for interactions . . . . . 24
2.2 The entities group . . . . . 25
2.3 The relationships between elements in entities group . . . . . 26
2.4 The services group . . . . . 27
2.5 The relationships between elements in services group . . . . . 27
2.6 The messages group . . . . . 28
2.7 The relationships between elements in messages group . . . . . 29

# List of Figures

2.1 The life cycle of a virtual enterprise . . . . . . . . . . . . . . . . . . . 14
2.2 The NIIIP system architecture . . . . . . . . . . . . . . . . . . . . . . 49

3.1 A Web services/multi-agents based model for enterprise collaboration 62
3.2 The internal architecture for an enterprise . . . . . . . . . . . . . . . 65
3.3 An agent-oriented UDDI architecture . . . . . . . . . . . . . . . . . . 80

4.1 A Web services / agent based system architecture for VE . . . . . . . 89
4.2 The internal system architecture for the customer entity (user agent) 90
4.3 The internal system architecture for an individual enterprise entity . 92
4.4 The internal system architecture for the Web portal entity (agent) . . 94
4.5 The sequence diagram for interaction among entities . . . . . . . . . 96
4.6 The system architecture . . . . . . . . . . . . . . . . . . . . . . . . . 99
4.7 The defined tasks in a bundle (call for bids) . . . . . . . . . . . . . . 102
4.8 The result of enterprises' bids . . . . . . . . . . . . . . . . . . . . . . 102
4.9 The JAX-RPC architecture . . . . . . . . . . . . . . . . . . . . . . . 104
4.10 The implementation architecture of the prototype . . . . . . . . . . . 106
4.11 The Web service for an enterprise . . . . . . . . . . . . . . . . . . . . 107

# Abbreviation

| | |
|---|---|
| ACC | Agent Communication Channel |
| ACL | Agent Communication Language |
| APD | Agent Platform Directory |
| ARB | Agent Resource Broker |
| CORBA | Common Object Request Broker Architecture |
| DAI | Distributed Artificial Intelligence |
| DAML | DARPA Agent Markup Language |
| DAML - OIL | DAML - Ontology Inference Layer |
| DAML-S | DAML-based Web Service Ontology |
| DARPA | Defence Advance Research Project Agency |
| DCOM | Distributed Component Object Model |
| DF | Directory Facilitator |
| DSOM | Distributed System Object Model |
| ebXML | electronic business XML |
| FIPA | Foundation for Intelligent Physical Agent |
| HTTP | Hyper Text Transfer Protocol |
| NCL | NIIIP Common Languages |
| NIIIP | National Industrial Information Infrastructure Protocols |
| OA | Ontology Agent |
| OMG | Object Management Group |
| ORB | Object Request Broker |
| PAL | Proxy Agents Layer |

| | |
|---|---|
| RMI | Remote Method Invocation |
| SOAP | Simple Object Access Protocol |
| UDDI | Universal, Description, Discovery, and Integration |
| VE | Virtual Enterprise |
| WfMC | Workflow Management Coalition |
| WSDL | Web Service Description Language |
| WSFL | Web Service Flow Languages |
| XML | Extensible Markup Language |

# Abstract

*Web Services* technology is an emerging paradigm in establishing a standard environment for legacy applications integration over the Internet. Using Web services and related technologies facilitate the implementation of a virtual enterprise across heterogeneous software platforms. However, Web services suffer from some shortcomings fulfilling requirements of setting up a reactive and autonomous collaboration among enterprises. The current technology of Web services registry, known as *UDDI*, is in its infancy stage.

The aim of this work is to integrate intelligent software agents and Web services in order to apply them to create a collaborative environment. We start with describing the concepts of virtual enterprise, Web services, and software agents as well as their requirements, problems, and benefits. To this extent, we identify the existing problems with Web services technology and discuss the feasibility of using software agents and their abilities to prevail those difficulties. This thesis proposes a Web services / agent-based model for both the internal architecture of an individual enterprise and the UDDI registry as well. We define a multi-agent model in different levels of enterprise's system architecture to accomplish a suitable selection of a registered service, to check the status of a process, to realize users' requests, and to react to them in a collaborative way with other agent-based Web services. Moreover, the thesis proposes a multi-agent model to define a dynamic workflow capable of coordinating and monitoring the processes defined in the workflows.

# Acknowledgements

When it comes to gratitude, it becomes clear that my part on fulfilling this thesis was less than others. I did it as a result of others' efforts.

I would like to thank Dr. Weiming Shen, my supervisor, for his many suggestions and constant support during this research. I am grateful to him because of his not only supervising but also kindness, and humanization. I am also thankful to Dr. Ruizhong Wei for his guidance through my graduate study at Lakehead University. His advice showed me the right path in my academic carrier. I must appreciate Dr. Hamada Ghenniwa, faculty of electrical and computing engineering at the University of Western Ontario for his many technical recommendations.

I had the pleasure of meeting nice faculties in computer science department at Lakehead University. I appreciate all of them and their kindness during my graduate study. Obviously, joining Lakehead University was one of the unforgettable events through my life that changed my way.

I should also appreciate the Integrated Manufacturing Technologies Institutes (IMTI) of National Research Council of Canada in London, Ontario, since provides me enormous facilities toward completing this thesis.

I am thankful to Dr. Hamid Khoosravi at the faculty of mathematics and computer science, University of Kerman, Iran, to help me preparing for graduate study.

I must also appreciate Ms. Roxana Ketabestan for her assist and encouragement to continue my education.

Of course, I wish to thank both my sisters to take care of family during the period of my study.

Finally, I am grateful to my parents for their patience and *love*. Without them this work would never have come into existence.

Thunder Bay, Ontario — Akbar Siami Namin
January 2004

# Chapter 1

# Introduction

The revolutionary approach of manufacturing systems design is changing from an isolated island of resources and data to a dynamic collaborative alliance of companies best known as a *virtual enterprise.* The virtual enterprise can be considered as a scenario that emerges in a world where individual companies come together as a coalition to fulfill the market demands.

*Web services* technology is an emerging paradigm in establishing a standard environment for legacy applications integration over the Internet. Using Web services and the related standards facilitate the implementation of a virtual enterprise in heterogeneous software platforms.

*Intelligent agents* are software entities, which are able to complete tasks on behalf of their owners autonomously. The power of self learning and reasoning allow an agent to be a suitable solution towards decision making with respect to circumstances.

In this thesis, we propose an integration of Web services and software agent technologies, to realize inter-enterprise collaboration while maintaining privacy. We believe that the concept of the virtual enterprise paradigm and the combination of the Web services standards and the unique characteristics of intelligent software agents are able to provide the most suitable solution for future Web-based manufacturing.

## 1.1 Global Competition and Enterprise Collaboration

Today's world business pushes enterprises to go away from an isolated of resources and data to a dynamic collaborative environment alliance of enterprises to carry out market demands and towards achieving customers' satisfactions. Enterprise coalition, either stable or transient, known as virtual enterprise [5], helps companies sharing their resources such as technologies, data, skills, and facilities and provides a new type of product or service, which is beyond the capability of any individual member of the alliance. By virtual enterprise paradigm any individual business firm, either small or large, can join into a business union to put forward its services and to take advantages of some resources that are provided by other members. However, prior to any operational stage, forming a virtual enterprise requests some vital prerequisites to be set. Basically, the following fundamental questions are subjected to be answered.

*How to realize customers' requests towards offering services to them?*

*How to locate suitable and reliable enterprises?*

*How to communicate with others?*

*How to monitor the progress of the delegated tasks?*

Business world changes frequently. In order to survive, small companies need to join together and form a coalition to be able to compete, create new kind of services, and satisfy market demands. Apparently, no individual enterprise can be on business stage without collaboration with others. The Internet and electronic business have affected the market deeply in such a way that economic growing depends on accessible resources over the Internet. It leads us to develop a new form of economic model such as *"virtual enterprise"*, where participants stack their resources to provide a new kind of service. A virtual enterprise yields some features, which are suitable to support business collaboration:

- Ability in creating new kinds of services for any enterprises regardless of their sizes and geographic locations.
- Enhancing the quality of services by sharing resources.
- Modularity and reusability of business processes.

Web services paradigm is an emerging technology that revolutionizes distributed computing by enabling communication at application level through the Internet regardless of platforms, applications, programming languages, data structures, and business models. A Web service is a self describing and a discoverable software component that assists applications to integrate and exchange business document messages in a *"service oriented"* approach. Web services technology, not only allows inter-operability among heterogeneous environments, but also provides a loosely coupled system architecture in which changing a component, at any level, has no impact on any other entity. Developed software services are searchable over the Internet by publishing in the *Universal, Description, Discovery, and Integration (UDDI)*, which addresses the business information of service providers as well as links to *Web Service Description Language (WSDL)* that provides some meta data for explaining services, methods, parameters, and returned values as well as data structures and user defined classes. A service requester, simply by binding to UDDI, discovers the desired services. Then, by applying the information sited in WSDL, they exchange business documents. Business-to-Business communication at application level facilitates the establishment of a temporary collaboration, which takes into account some crucial parameters such as cost and time. A Web service offers some facilities that make it a superior selection in creating a collaborative environment.

The primitive combination of component-based middle-ware such as COM/DCOM, CORBA, and Web technologies to integrate business processes seems good but not complete. The combination suffers from the ability of integrating different data models, workflows, and business rules. Therefore, the main demand is to have a new technology capable of integrating heterogeneous data models, and business rules. This is the essence and power of new paradigm known as *"Web services"*.

Web services have some similar features to the component based approaches like CORBA and COM. However, there exist some differences between the two technologies [24]:

- Web services technology uses a "document based" communication, however component based approach adopts an "object-based" one. This feature makes Web services technology as an independent software , or loosely coupled component. In contrast, object based nature of component based technology implies dependencies among entities or *"tightly coupled"* systems.

- The transportation protocol for component-based technology is IIOP, which is not able to traverse firewalls easily. By employing HTTP as a communication protocol in Web services, passing documents throughout firewalls is easier.

In IT industry, Web-based technologies are ongoing approaches, but also have raised some problems. The main problem is:

*How to unify heterogeneous data models in different platforms to produce an integrated solution?*

*"Web services"* technology seems being able to solve some of the problems concerning the Web based paradigm. The main objectives in distributed computing are: universal connectivity, decentralization, openness and dynamic evolution [1]. Web services technology is a new equipment to achieve some of these goals by proposing *"software as a service"*.

Web services are believed to be capable of revolutionizing distribution computing by enabling inter-operability at such levels, which have never been achieved. The main advantages of Web services approach can be considered as:

- *Inter-Operability.* The previous generation of distributed computing such as CORBA and COM/DCOM, due to the lack of uniform standards, had some limitations. Web services are implementable by any language and platform and are intractable with any other Web services deployed on different environments.

- *Minimum Cost.* Simplicity of implementation and universal supports for Web services to be a standard, have made it an affordable technology even for small companies.

- *Vendor Support.* Web services technology is a product of some main software vendors companies such as Microsoft, IBM, HP, etc. On the contrary with other software technologies, Web services technology has been supported by industry from the beginning.

However, despite offering revolutionary technologies in Business-to-Business co-operation and feasibility of applying composite services to a dynamic environment such as virtual enterprise, Web services suffer from some shortcomings in fulfilling all requirements of setting up a reactive collaboration. These problems can be viewed at any level of adopting Web services in a virtual enterprise. The current technology of Web services standard in publishing, locating, and choosing a service in the *Universal, Discovery, Description, and Integration (UDDI)* is still in its infancy stages. Service requesters do not have any clue about *reputation, qualification, history, and ranking* of service providers to assist them in choosing the most suitable business partners. Moreover, the current technology of UDDI is based on *"syntactical"* matching rather than *"semantically"* equivalency. Indeed this kind of searching mechanism is not enough for service both providers and requesters.

The current paradigm in offering, invoking, and communicating among Web services is not able to handle and realize all incoming and outgoing messages between two parties. The contents of *Web Service Description Language (WSDL)* are fitted for *"invoking"* methods rather than *"requesting"* them. In fact, a service provider should be highly intelligent to realize the customer requirements and carry out them in a reasonable way.

Software agents are a new paradigm derived from *"Distributed Artificial Intelligence"*, which has been proposed as a suitable technology to create a cooperative intelligent environment. There are some general reasons in choosing software agents as solutions for business models [16]:

- *"Autonomy"* of an agent contributes autonomy for components involved in a process and consequently in the workflow.

- Agents are *"social"*, which means they are capable of collaboration, negotiation, and coordination, which are known as interaction among components in a business process.

- Agents are *"proactive"*, which addresses the responsibility of components for reacting to changes in environment pro-actively.

These properties make agents technology as a suitable paradigm in fulfilling business processes.

## 1.2 Scope of the Thesis

Although we are dealing with three important major concepts, we do not go deep inside each of them. The scope of this thesis is mainly about the integration of two concepts: Web services, and intelligent software agents towards setting up a virtual enterprise.

The thesis focuses on the relationship among three concepts and how to use them to set up a collaborative environment. We look at virtual enterprise as a major concept and use Web services technology to establish a virtual environment. Furthermore, we discover that Web services paradigm is not capable of handling all requirements of setting up a collaborative environment, thus we adopt intelligent software agents to overcome those shortcomings of Web services on preparing a suitable environment for a virtual enterprise.

The work first reviews general concepts of the underlined related technologies and then identifies the requirements or circumstances by which a collaborative environment takes place. Then the thesis proposes a multi-agent based approach to Web services paradigm to realizing service requesters' requirements. Software agents are defined at each layer of an enterprise. To this extent, some agent based units are created either inside the enterprise or *Universal, Discovery, Description, and Integration (UDDI)* registers.

On the enterprise side, a multi-agent model has been proposed, which consists of some units such as *"Workflow Designer", "Controller", and "Coordinator"*. The thesis argues the needs for each unit to provide solutions for some of the primary questions. Moreover, the thesis proposes a multi-agent model for the UDDI registry, which is not only capable of reasoning to assist service requestors towards semantically searching, but also offers some valuable data to help customers in selecting the desired service providers with respect to their *"reputation", and "ranking"*.

## 1.3 Organization of the Thesis

We reviewed overall requirements for enterprise collaboration and how Web services technology offers some facilities to carry out some of the requirements. In Chapter 2 we review the major concepts; some related projects as well as other published results. We compare Web services and software agents to find out if there exist any similarities between them. We give a brief explanation of their strengths and weaknesses. Moreover, we clarify the exiting problems with them and how we are going to solve some of them. In Chapter 3 we propose a multi-agent based model for Web services to overcome some shortcomings of Web services in order to form a collaborative environment. We identify the needs for using agents in each layer of Web services. Moreover, we adopt our multi-agent model both in enterprise and service registries (UDDI). Our developed prototype architecture and implementation are discussed in

Chapter 4. We have prepared a prototype to discover the shortcomings and advantages of Web services by practice. The prototype shows how to connect, deliver, and return data to and from a Web service. In Chapter 5 we provide a brief conclusion and discuss the future work. Our conclusion and the enormous listing of future works show that Web services technology is still in its infancy stage.

# Chapter 2

# Literature Review

In this chapter we review the major concepts used in this thesis and the related literature. We highlight the problems exiting in the literature in order to clarify our approach in proposing a multi-agent model in the next chapter. Furthermore, we compare the Web services and software agent technologies to represent the similarities between them.

## 2.1 Concepts Review

In this section, we briefly review the concepts of Virtual Enterprise, Web Services, and Software Agents, which are the main focus of the thesis.

### 2.1.1 Virtual Enterprise

At this age of global economy, no any individual enterprise can survive independently. World class business demands communication and resources sharing among enterprises not only to reduce cost and time, but also to create a new kind of services. Collaboration for small and medium size enterprises is a mandatory issue towards sharing their skills and resources for competition purposes. In recent years, Web based technologies have brought a new paradigm of collaboration and resources sharing applicable throughout the Internet. Nowadays, online trading over the Internet is a daily process in any kind of businesses. The requirements for resources sharing have introduced the concept of *"Virtual Enterprise."*

The coalition and communication among members of a VE, in terms of software, are carried out by computer networks, applications and software technologies.

The *"Requests for Services"* raises the needs for collaboration. If both requests and services are predefined and static, the collaboration will be a permanent interaction. On the contrary, in dynamic cases of both requests and services, the collaboration is treated as temporary one. Requests vary because enterprises have diverse needs. As a result, service variations may occur for any technical reason. To accommodate a dynamic property, the process of assigning services to requests must be dynamic [25]. This idea is the essence of the concept of virtual enterprises. A Virtual Enterprise is a framework for dynamic changes for both requirements and services. Assignment of available services to requests can be viewed as many-to-many mapping that change over time.

**Definition 2.1.1. Virtual Enterprise.**

*A temporary goal-oriented cooperation and alliance of enterprises to share resources and produce a new product or service for customers such that no any individual enterprise can produce the same product or service independently.* □

Moreover, according to [13] a virtual enterprise is defined as: *"Virtually (temporary) united of independent member companies to exploit a world wide business opportunity. A consortium of enterprises manufacture products together, none could* *build alone"*. A VE consists of several workflows and each workflow is composed of several processes.

Networking inside a VE is supported by information and communication technology. The virtual characteristic of a VE makes it free from either organization chart or hierarchy. A virtual enterprise can have centralized or decentralized control workflows, structures, and functions. The shared resources compose of wide variety of objects such as: skills, experienced workers, physical machines and devices, software systems, databases, etc. At the end of the life cycle of a VE, the involved companies, based on circumstances, decide to continue or dissolve the VE. We may consider a VE as a supply chain management system in which each individual enterprise behaves like a node that adds some values to the entire supply chain. There exists some other related terms besides the VE as: Extended Enterprise, Virtual Organization, Networked Organization, Supply Chain Management, Cluster of Enterprises, etc. There exist some similarities as well as dissimilarities among them which are out of scope of this thesis. Interested readers are suggested to refer to [5].

The temporary nature of a VE leads us to define a formal life cycle for it. There is a basic life cycle introduced by [5] including: *"Creation"*, *"Operation"*, *"Evolution"*, and *"Dissolution"*:

1. *Creation.* This stage starts when an individual enterprise wants to handle a customer's requirements and is unable to do them independently. There is some other functionality included in this stage as: partner's selection, negotiation, privacy and sharing level, job description, coordination, etc.

2. *Operation.* At this stage the enterprises who are involved in the VE work on their assigned tasks. They exchange information and share resources with

respect to security concerns and privacy issues, which have been set during the creation stage of the VE.

3. *Evolution.* Not every coalition of enterprises is perfect. Changing is one of the major challenging issues in every system. In any VE, regarding policies, adding or changing a business partner is a natural action.

4. *Dissolution.* This is the final stage of the life cycle, at which the VE results in success or failure. At the final step of a VE, enterprises decide to continue or dissolve the VE. The final stage is more than just dissolution. At this stage, the enterprises involved in a VE learn from the past stages and use them in future's possible coalitions.

At the creation stage of any VE, it is necessary for enterprises to choose one as a *"coordinator"*. A coordinator can be either a member of coalition or an external enterprise. Scheduling and monitoring are amongst important duties for a coordinator. The other responsibilities of a coordinator are:

- Looking for business partners.
- Supervising all stages through the life cycle of a VE.
- Monitoring all enterprises' activities regarding the VE's goals.

We depict the processes, which are defined through the life cycle of a VE in Figure 2.1.

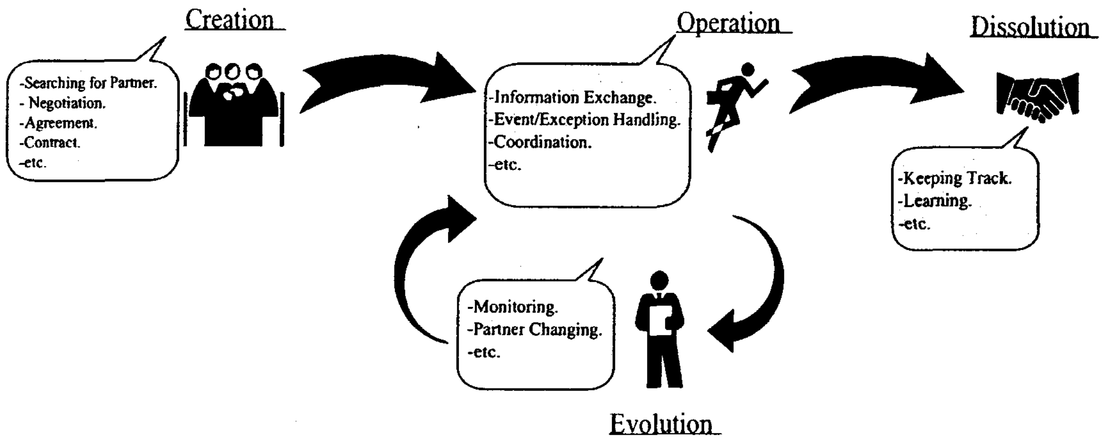


Figure 2.1: The life cycle of a virtual enterprise

### 2.1.2 Web Services

Web services, by using standard communication protocols, allow applications on different platforms and environments to communicate and exchange data over the Internet. The main purpose of Web services is for distributed computing, which emerges to perform tasks in a distributed way rather than a centralized one [10]. Encapsulation and hiding complexity of components are among the first benefits of using this technology in distributed systems. Component technology, which has been used in distributed computing very widely, considers every datum, event, or human operator as an object. In recent years, many technologies have been developed in terms of components and distributed computing, such as OMG's product (Object Management Group), Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM), Sun Micro System's Remote Method

Invocation (RMI), and IBM's Distributed System Object Model (DSOM). However, despite some advantages of these technologies in distributed computing, they have some limitations in communication and resources sharing. Due to the diversity of platforms and operating systems in different enterprises, communication without considering some universal standards is impossible. The Web services technology improves distributed computing by enabling communication at the application level.

The authors at [1] have defined Web services as: *"Internet-based applications fulfilling a specific task or a set of tasks that can be combined with other Web services to maintain workflow or business transactions".*

Web services are software components, which use XML, a text based standard format, to communicate with other software components, either at the same platform or on different environments, through the Internet. The main purpose of Web services is to call a remote procedure and ask for an operation *(request / receive action).* The most popular protocol for exchanging over the Internet, or the *"Hyper Text Transfer Protocol (HTTP)"* is the underlined communication protocol in Web services technology. Web services have some other characteristics:

- **Encapsulation.** A Web service hides complexity of the tasks. Requesters just submit their inputs and receive the results from Web services. It means that a Web service is implementable. The *"sharing process"* is a key concept in Web services, which is the result of encapsulation of services by just invoking them through their interfaces.

- **Text Based Inter-operation, XML.** The text based property enables different applications in heterogonous environments to communicate regardless of platforms, data structures, and data models.

- **Self Describing.** A Web service is capable of describing itself, results, facilities, and using methods. This information has been provided by the *"Web Service Description Languages (WSDL)"*, a text based XML standard.

- **Easy to Search.** A Web service may register and interested service requesters may find its location by searching the registry. The search is possible through the *"Universal Description, Discovery, and Integration, (UDDI)"*, a text based XML standard.

In order to exchange understandable messages, all parties should use some universal rules and data structures. The standardization is possible through developing some globally known protocols at different levels. In Web services technology there exist three standard protocols as SOAP, WSDL, and UDDI, which are based on XML standard. In this section we briefly introduce them and their functionalities.

**SOAP (Simple Object Access Protocol).**

The *Simple Object Access Protocol (SOAP)*, is a messaging framework and software component, which allows application to communicate and exchange data. Each document, which is subject to transmit, is called *"SOAP message"*. SOAP messages are not operational; rather they specify which operation should be called. On other words, SOAP is a framework to describe programming actions and behaviors.

**WSDL (Web Service Description Language).**

In order to submit queries to a Web service, a SOAP client must have enough information regarding the Web service, its description, and functionalities. A client must know how to connect, invoke, and receive results from an individual Web service.

This kind of information for each Web service is stored in *Web Service Description Language (WSDL)* documents. Locating these kinds of documents is possible via registry entry as UDDI. WSDL is XML-based, which provides some meta-data for using an individual Web service.

### UDDI (Universal Description, Discovery and Integration).

In order to set up a virtual enterprise, individual enterprises need an easy way to locate desired business partners or services in a more structural way. One way is to publish information about any services by service provider itself and on its own side. However, searching and locating all enterprises and their provided services through the Internet is a time consuming and absolutely not a convenient task. In order to facilitate locating proper services or companies, the *Universal Description, Discovery, and Integration (UDDI)* standard has been proposed, by which each enterprise interested in offering some sort of products or services, register their services and some other information regarding the Web services for discovery purposes by others. Basically the registry acts like a repository for some data that describe business capabilities. An enterprise, by using registry, not only finds out desired services more easily, but also compares similar offered services in terms of their types by different enterprises and chooses the most appropriate one, which fits the enterprise needs.

### Service Composition.

One of the most important results of employing Web services is using the combination of Web services to achieve new kinds of services. Some researchers believe that the entire value of Web services is in their ability to define service composition. Composite services, or value added services, are created by stacking skills and resources, and are treated as new services. By composing some services, enterprises can form an alliance to provide *"One-Stop Shopping"* for customers [2].

A composite service can be viewed as a service provider that has its own WSDL documents and is accessible for clients to use the composite service. A composite service provider may also register its services through the UDDI registry. The aggregation makes the Web services, which are involved in composition, to collaborate with each other. Due to the inter-dependency among Web services and their own flows of control, the aggregated services need some attentions in defining both control and data flow.

### 2.1.3 Software Agents

Classical artificial intelligence mainly focuses on intelligent reasoning and problem solving. The idea of distributed problem solving and distributed computing has introduced the *Distributed Artificial Intelligence (DAI)*. DAI is concerned with distribution of intelligent processes among independent entities [31]. An entity in DAI is defined as an *agent*.

There have been some definitions for software agent [31]. Brenner [3] defines a software agent as:

*"A software program that can perform specific tasks for a user and has a degree of intelligence that permits it to perform parts of its tasks autonomously and to interact with its environment in a useful manner".*

Jennings and Wooldridge [48] define an intelligent software agent as:

*"A computer system that is capable of flexible autonomous action in order to meet*

*its design objectives."*

Among properties, which have been considered for an agent the following are more important:

- *Autonomous.* An agent should be capable of working independently without being controlled by another entity.
- *Social.* An agent is not an isolated component of information and abilities. Agents *must* interact and exchange information with others. In fact, any software components without any interaction with other components can not be considered as an agent.
- *Reactive.* Reacting to changes in the environment is one of the most important properties for an entity to involve in a dynamic collaboration.
- *Proactive.* Pro-activeness is one of the main characteristics of agents, which allows agents to take initiatives and prepare for next stages.

In a virtual enterprise, components need to communicate with each other. An agent can take place as a component in a VE. In fact, there exists a multi-agent environment to handle business processes. In multi-agent paradigm, each agent has been planned to acquire its internal goals such that whenever a set of agents collaborate, they contribute to achieve the *"common"* goals.

**Communication.** The main issue in designing a multi-agent environment is communication among agents. Communication means exchanging information and is essential for coordination and cooperation. The communication method depends on the type of agents. For instance, communication takes place if an agent executes some rules. The activation and execution of the rules depend on behaviors and reactions of agents to changes in the environment. Thus, by executing these rules, the agent sends some signals or data to destination for submitting or asking for information.

**Cooperation.** According to Doran [11], cooperation is defined as:

*"In a multi-agent environment, cooperation occurs when the action of each agent satisfying either or both of the following conditions:*

1. *The agents have some goals to achieve such that no achievement is possible in an isolated model.*
2. *The agents fulfill actions to achieve either their goals or others."*

There exist three levels of cooperation [31]:

- *Fully Cooperation.*
- *Partly Cooperation.*
- *Non - Cooperation.*

In fully cooperation, each agent adopts its private goals to achieve the common goals, which are known through all agents. On other words, agents are interested in achieving overall goals rather than private goals. However, in a non-cooperation case, agents act as stubborn components, which do not like changing their private goals to contribute in achieving the common goals.

**Coordination.** Jennings [14] has defined coordination as:

*"The process by which an agent reasons about its local actions and the actions of others to try and ensure the community acts in a coherent manner."*

Dependency among agents, especially in their goals, and achieving a common result, makes coordination as an essential part for multi-agent systems. For instance, in a virtual enterprise, common goal is to offer some products to customers, which no any enterprise is able to produce it independently. Thus, the output of each individual enterprise contributes toward achieving the final product.

### 2.1.4 Web-Services vs. Software Agents: A Comparison

In this section, as a research result, which we have encountered during the preparation of the thesis, we compare Web services and software agents to discover the possible similarities between them. On other words, we are interested if it is possible to view a Web service as an agent. The comparison will be based on the FIPA specification for software agents.

#### Is a Web Service An Agent?

As an interesting point, we are concerned about if Web services technology satisfies the requirements of software agents' specification. The system architectures for both technologies look very similar. However, we must consider a standard comparison between two technologies. So, the challenge is:

*"Is it possible to view each Web service as an agent?"*.

In order to have a standard and reasonable comparison, we focus on the standard specification of FIPA [43].

An agent is an active object, which is not controlled by another entity. Wooldridge and Jennings [48] have divided agents into:

- *Weak.* An agent is weak if it is autonomous, social, reactive, and proactive.

- *Strong.* An agent is strong if in addition to the properties of a weak agent; it has one or more of the following: mentalities notions (beliefs, goals, plans, and intentions), adaptability, etc.

In fact, an agent is a software abstraction with behavior, inheritance, encapsulation, state, and identity and a strong agent is an active object that is able to reason and communicate via structured messages [43]. Generally, whenever there is a reference to an agent, the strong one is considered. However, an object with some characteristics can be viewed as a weak agent as well. A strong agent has common sense mental structure, beliefs, desires, and intentions. However, a weak agent is an object with some agenda. So, obviously *an independent functioning unit*, such as an object, can be viewed as a weak agent. Therefore, a Web service, as an independent functioning unit (not controlled by another entity), with some agenda (offering some services) can be treated as a weak agent.

FIPA defines an agent as [43]:

*"An agent is a computational process that implements the autonomous, communicating functionality of an application".*

or in detail:

*"An agent is an encapsulated software entity with its own state, behavior, thread of controls and ability to interact and communicate with other entities, including people, other agents, and legacy systems."*

Now, we try to describe a Web service in terms of the first definition of an agent:

*"A Web service is a computational process (*method and procedure*) that implements (*implementation part*) the autonomous (*not being controlled by another Web

service*), communicating (*throughout SOAP messages*) functionalities of an application (*service*)"*.

Based on the second definition of an agent, we describe a Web service as:

A Web service is an encapsulated software entity (*by interface*) with its own state (*some rules*), behaviors (*reaction to requests*), and thread of control (*internal workflow*) and an ability to interact and communicate (*through SOAP messages*) with other entities including people (*client request*), other agents (*other Web services*), and legacy systems (*regardless of platforms*).

Until now we are able to define a Web service in terms of the agent terminologies.

FIPA lists some requirements for interactions between an agent and an agent, an agent and a user, as well as an agent and an existing information source. However, the point is that not all behaviors are presented in a particular agent [43]. Based on the FIPA specification, in Table 2.1, we compare some of the basic requirements for interactions in both agents and Web services technologies.

The FIPA abstract architecture [44] defines a set of architectural elements and their relationships to define, locate, and communicate with each other. We first look at the definition of elements in agent technology based on the FIPA specification and then try to find an equivalent element in Web services technology with similar functionalities. To this extent, we focus on the existing defined relationships among elements in FIPA specification and our effort to find the same relations in Web services technology.

We have divided the elements, which have been defined in FIPA specification [44] into these groups: Entities, Services, and Messages groups. We review and compare the elements of Agents and Web services in each of these groups as well as their

Table 2.1: Agents and Web services corresponding basic requirements for interactions

| **Software Agents Technology** | **Web Services Technology** |
|---|---|
| An agent needs to publish its services and resources to other agents as well as discovering services or resources of other agents [43]. This is achievable through registration in the Directory Facilitator (DF). | A Web service publishes its services or resources in order to be discovered and used by other Web services or clients. This is achievable through registration in the UDDI registry. |
| Agents need to communicate with each other. They should also be able to use existing infrastructure (TCP/IP networking, HTTP, etc) [43]. | Web services communicate with each other through SOAP messages standard. The transportation protocol of Web services is HTTP (regardless of platforms). |
| Agents need to be able to access (existing) information sources[43]. | Web services, as long as the XML based SOAP messages transportation is possible, can access to existing information sources. |
| The communication should be robust in the sense that recovery from exception conditions should be possible [43]. | An asynchronous communication is robust. In the synchronous case, which a Web service needs some information from another one, it suspends the execution until receiving information. However, a Web service is multi-threats and offers services to a large number of users at the same time. |
| Security issues of offering services. | Internal policies of Web services. |

Table 2.2: The entities group

| Elements Description | Agents Tech. | Web Services Tech. |
|---|---|---|
| A computational process that implements the autonomous, communicating functionality of an application. | Agent | Web service |
| An entity consists of: name, locator, and attributes. | Agent Directory Entry | Included in UDDI |
| Providing some meta data regarding offered services. | Agent Directory Service | Included in UDDI |
| A set of transport-descriptions used to communication. | Agent Locator | A link to WSDL in UDDI |
| Unique identification. | Agent-Name | Web service name |

relationships separately.

- Entities Group. Table 2.2 shows the elements of this group and their corresponding elements in the Web services technology. Table 2.3 demonstrates the relationships among the existing elements in the Entities Group.

- Services Group.

  Table 2.4 shows the elements of this group and their corresponding elements in the Web services technology. Table 2.5 demonstrates the relationships among the existing elements in the Services Group.

- Messages Group.

  Table 2.6 shows the elements of this group and their corresponding elements in the Web services technology. Table 2.7 demonstrates the relationships among the existing elements in the Messages Group.

Table 2.3: The relationships between elements in entities group

| **Relationship** | **Agents Tech.** | **Web Services Tech.** |
|---|---|---|
| Each entity has a name. | Agent-Name | Web Service Name |
| Entities communicate via: | ACL | SOAP |
| Provides a location where entities register their descriptions. | Agent-Directory-Service | Included in UDDI |
| Other entities search to find out services. | Agent-Directory-Entities | UDDI |
| Registering is by saving: | (Agent-Name, Agent-Locator) | (Some Meta-data, A Link to the WSDL) |
| Registering and discovering are achievable by: | Agent-Directory-Entry | UDDI |

We have constructed a corresponding mapping between some elements of both agents and Web services. Also, we showed that even the relationships among elements in both technologies are very similar. In fact, both agents and Web services not only communicate through their languages but also their processes of creating, deploying, registering, and discovering are very similar.

### Software Integration Comparison

Now, we go further and compare the two technologies at a higher level. In order to use a non-agent software system FIPA specification [45] identifies two agent roles:

- *Agent Resource Broker (ARB).* An ARB agent brokers a set of software description to interested agents. Clients query it about what software services are available.
- *WRAPPER Agent.* This agent allows an agent to connect to a software system uniquely identified by a software description.

Table 2.4: The services group

| Description | Agents Tech. | Web Services Tech. |
|---|---|---|
| Provided for entities and other services. | Service | Service |
| Specific string indicating how to bind to a particular service. | Service-Address | Included in WSDL |
| An entry for saving some information about service. | Service-Directory-Entry | UDDI entry |
| For registration and locating services. | Service-Directory-Service | UDDI registry |
| Provides unique presentation for service. | Service-Name | Service Name |
| A parallel structure describing how to access a service. | Service-Location-Description | Include in WSDL |
| Can be used to access and make use of a service. | Service-Locator | A link to WSDL in UDDI |
| Describing the binding signature for a service. | Service-Signature | Security issues in Web services |
| Defining the type of a service. | Service-Type | Included in UDDI |
| Describing the type of a service signature. | Signature-Type | Security issues in Web services |

Table 2.5: The relationships between elements in services group

| Relationship | Agents Tech. | Web Services Tech. |
|---|---|---|
| Provides a location where services can register their services descriptions. | Service-Directory-Service | UDDI database |
| Are able to search | Agents | Web services |
| (Service-Directory-Entry or UDDI) is a: | Key-Value-Type | XML based |
| (Service-Directory-Entry or UDDI) consists of: | (Service-Name, Service-Type, Service-Locator) | (Service-Name, Meta-data, A link to WSDL) |

Table 2.6: The messages group

| **Description** | **Agents Tech.** | **Web Services Tech.** |
|---|---|---|
| A unit of communication between two entities. | Message | SOAP message |
| Information about message delivery and encoding. | Envelope | SOAP envelope |
| Encodes the structure of a message. | Encoding-Service | Data types encoding rules in SOAP |
| Supports sending and receiving (transfer) between entities. | Message-Transport-Service | Binding framework in SOAP |
| An encoded message supposed to be sent (actual data). | Payload | SOAP payload |
| A language basis of communication between entities. | ACL | SOAP |
| The part of a message, representing the domain of communication. | Content | SOAP body |
| A language used to express the content of a communication. | Content Language (FIPA-SL, FIPA-RDF) | XML |
| A way of representing a message. | Encoding-Representation (XML) | XML |
| A particular message delivery. | Transport | Binding framework in SOAP |
| The object conveyed from entity to entity. | Transport-Message | SOAP envelop |
| A self describing structure to describe how to deliver the message. | Transport-Specific-Address | WSDL |
| An address specific to a particular type. | Transport-Type | Included in WSDL |
| The type of transport. | Transport-Description | Included in WSDL |

Table 2.7: The relationships between elements in messages group

| **Relationships** | **Agents Tech.** | **Web Services Tech.** |
|---|---|---|
| Structure of message | Key-Value-Tuple | XML based |
| Written in | ACL | SOAP |
| Content of the message is expressed in: | Content-Language | XML |
| A message contain: | (Sender, Receiver) | (Sender, Receiver) |
| A message is encoded into: | Payload | Payload |
| A message is included in: | Transport-Message | SOAP envelop |
| The payload is encoded using: | Encoding-Representation | XML and encoding rules in SOAP. |
| A transport consists of: | (Payload, Envelop) | (Envelop) |
| Each entity has one or more: | Transport-Description | WSDL |
| A set of Transport-Descriptions can be held in: | Agent-Locator | Links to the WSDL in UDDI |

Basically, a client agent, in order to access a non-agent software system, queries the ARB and finds out the WRAPPER agent that is responsible for the desired software system. FIPA specification in software integration [45] has defined a reference model, which consists of:

- *Directory Facilitator (DF).* A specialized agent, which provides a *"yellow page"* directory service. Agents advertise their services to an agent domain by registering with the DF.

- *Agent Communication Channel (ACC).* Providing a message-routing function for inter-agent communication (Inter-operability).

- *Software (SW).* Non-agent software systems are described by software descriptions.

- *ARB agent.*
- *WRAPPER agent.*

In the reference model, the agents, which represent ARB and WRAPPER register with the DF. In fact, by registering with the DF, they advertise that they provide software description for both ARB and WRAPPER services to clients. Any agent interested in using a registered non-agent software system queries the DF for an agent, which provides an ARB service. Furthermore, The agent by finding and querying the ARB agent asks for a software system, which matches some specific requirements and the ARB agent returns a software description, which uniquely identifies a specific software service. Finally, the client agent queries the DF to find a WRAPPER agent for the desired software service and starts to communicate with the WRAPPER agent.

By comparing the software integration methods using agents and Web services, we find out that there exists a similar scenario in the Web services technology as well.

In Web services technology the UDDI registry can play the role of Agent Resource Broker (ARB). In fact, the UDDI brokers a set of service description to interested users. Clients by querying the UDDI find out what kind of software services are available.

Each Web service has both interface and implementation. In terms of agent technology, the service implementation has the role of software system and the service interface plays the role of WRAPPER agent. The service interface, or WRAPPER agent allows a client to connect to a software system. Also, the UDDI registry, can play the role of *Directory Facilitator (DF)* in agent paradigm as well.

In Web services paradigm services register in UDDI, (*or DF*), and any client, (*or agent*), interested to find a software system queries the UDDI, (*or DF*). Consequently, the UDDI returns the service, which offers the desired software system. Simply the client, (*or agent*), queries the interface of service, (*or WRAPPER*), to use the software system.

This section discussed the similarities between Web services and software agents technologies. The comparison was based on FIPA specification for software agents. By constructing a mapping between some elements in both agents paradigm and Web services technology and also their corresponding relationships, we demonstrated that: Indeed, the Web services technology has many similarities with the agents paradigm.

## 2.2 Literature Review

Virtual enterprise based on agents and Web services is a new research areas in computer science related fields. There have been lots of efforts and researches to integrate software agents in a VE. By the emergence of Web services, it seems researchers are focusing on adopting Web services and software agents together towards fulfilling the requirements of enterprise collaboration. There have been a few efforts to use agents in Web services, but each of them addresses some specific usages of agent's properties in Web services. We have classified the related researches based on their approaches to *workflows*, composite services, quality of services, and finally general adopting of agents to Web services technology.

### 2.2.1 Web Services/Agent-based Approaches for Workflows

Computerized workflow has been one of the main research area in software engineering for a long time. There have been some approaches for defining a workflow in terms of the used technologies. In this section, we are interested just in those cases, which have been defined in terms of software agents and/or Web services.

### Agent-based Approaches

The use of agents in workflow management systems, WfMS, has been discussed in several publications. In [7], each workflow has been represented by multiple agents:

- *"Personal Agents"* act on behalf of actual participants. A personal agent represents an actual individual member in a business organization with multiple roles.

- *"Actor Agents"* is viewed as a specific instance of a personal agent with a certain role such as object modeler. A mobile actor agent has a list of goals, tasks and a process in which a specific role is involved.

- *"Authorization Agents"* work as facilitators for a set of specialized functions such as workflow, communication. Authorization agents know how to retrieve valuable information, either from existing tools or from their own repository.

These agents act as personal assistants performing actions on behalf of the workflow participants and facilitating interaction with other participants or organization.

In [15], the *"Advanced Decision Environment for Process Tasks (ADEPT)"* project has been described. The multi-agent architecture consists of a number of autonomous *agencies.* A single agency consists of a set of subsiding agencies, which are controlled by one responsible agent. Each agent is able to perform one or more services. These atomic agents can be combined to form complex services by adding *"ordering constraints"* and *"conditional control"*.

None of [7] or [15] adopts agent technology to compose workflow execution engine dynamically. The logic for workflow processing is hard-coded and thus it is hard to reuse the workflow execution engine for other business process.

In [28], a *"Multi-Plan State Machine"* agent model has been presented. Agents are assembled dynamically from the description into a language and can be modified while running. But users have to describe, in detail, how agents can be assembled together and what changes are allowed.

In [50], the use of agents to integrate *"cross-enterprise"* dynamic workflow has been discussed. Three components have been defined:

- *Workflow Definition Tool.* Users are responsible for defining cross-enterprise workflows specifications. It allows a user to chart a personalized cross-enterprise workflow using a Web Browser based on existing or new templates.

- *Agent Society.* A collection of agents that provide the general functionalities of a cross-enterprise workflow execution engine. The agents have been categorized into three classes:

  1. *Process Agent.* Responsible for transforming a workflow specification into a workflow instance.
  2. *Discovery Agent.* Responsible for assigning tasks to some entities.
  3. *Monitor Agent.* A mobile agent responsible for monitoring the execution of a given task at a local host, where the corresponding task is executed. It is capable of informing the process agent or other monitor agents of the execution status of a task.

- *Actual Services.* A collection of applications, which offer services that allow users to access and perform tasks.

The model in [50] can be viewed as a great proposed idea to utilize software agents in defining workflows. However, the following points are questionable in this model:

1. The definition of a workflow is based on the traditional definition of workflow in which a workflow consists of some tasks and each task is supposed to be executed with respect to the *Event-Condition-Action* rules. This kind of *"predefine"* definitions of rules, is the same as static definition of workflows in which the rules are hard-coded and are not changeable even at runtime.

2. After defining a workflow at the design time, the designer submits it to the agent society to carry out. So, despite the availability of some templates, another shortcoming of [50] is defining a workflow initially by human users.

3. The monitoring of the execution of tasks is just handled by a monitor agent, which migrates to the local host. There is no defined mechanism to compare the plan and the execution process and inform users or agent society to change or cancel the process.

4. The model does not address the *"dynamic"* locating of a *"suitable"* partner and it is supposed to have a *"well known"* business partner. Furthermore, there is no mechanism to assign tasks to partners.

### Web Service/Agent based Approaches

OASIS proposed *"Business Transaction Protocol"* [6], a specification for an XML based protocol for managing transactions over the Internet. The business transport protocol incorporates a two-phase commit protocol to control workflows. It has been designed to allow coordination of applications works between multiple participants owned or controlled by autonomous organizations.

A *"Shadow Board Agent"* architecture by orchestrating multiple Web services in an agent based transaction model has been proposed in [17]. Each participant has been wrapped as a Web service and uses an agent oriented approach to engineer each Web service as a software agent. The model specifies the common communication protocol between multiple agents and lets subagents deciding about their own internal transaction process. Consequently, the internal behavior and control logic is isolated from the overall transaction level.

At the center of the whole transaction agent [17], the *"Coordination and Delegation Agent"*, or *"Root"*, has been sited, which is the only contact point to the requester user applications. The Root engages a number of *"Resource Agents"* for coordinating resources and *"Wrapping Agents"* for Web services. Moreover, the Root is associated with a *"Discovery"* in order to discover the services. The model proposed in [17] focuses on how to perform transactions in loosely coupled environments.

Despite a great agent-based architecture, the model can be criticized as:

1. The initial defining of workflow is not discussed in the model. It does not address how to generate a workflow and who is responsible for it.

2. The model does not address monitoring mechanism. The assigning tasks to participants are one way action and there is no control on checking status of the delegated tasks and probably replacing the existing partners.

3. The rating mechanism is not suitable. The information regarding the qualification of an individual partner is saved at the Root component and is not available for public access. Consequently, In the case of needs to a new partner, the rating system in this model cannot answer which candidate is more suitable.

4. Despite claiming the elimination of central coordination in fulfilling jobs, there is a need to the Root component to coordinate the whole tasks in the workflow. Thus, in fact there is a need to a central coordination for fulfilling the related tasks in workflows.

There are some other research papers related to using agents and Web services for defining workflows [19] [4], and [35]. Most of them address a high level conceptual description of topic such as defining a multi-layer business process or using agent in a very ambiguous way to search directory registry. They have not provided a detailed description regarding adopting agent paradigm and Web services to create workflows.

### 2.2.2 Agent-based Approaches for Service Composition

Web services are most valuable when composed together. However, existing approaches are limited to a hard-coded procedural abstraction for composition and cannot handle flexible interactions to support realistic composition. There exist some researches to propose a dynamic composition based on agents.

An interesting conceptual and generic approach has been proposed in [33]. The approach is based on *"temporal logic"*, which has rigorous semantics, and yields a naturally distributed execution. A multi-agent system has been taken on the problem of service compositions, emphasizing the agent based distributed computing. The approach is to achieve service composition by using an agent for each service to be composed and declaratively specifying workflows for the agents. The agents can dynamically and with maximum flexibility handle the right events with enough constraints to satisfy the workflow. It can facilitate the design and enactment of co-ordinated behaviors by hiding low-level details. By separating the specifications and the internal languages, it is possible to declare specification and apply operational decision procedures for them.

Agents act autonomously, constrained only their coordination relationships [33]. Thus decisions on events can be taken on local information. Moreover, decision-making functionality occurs when the decision needs to be made. In [33], there has been defined a *"guard"* on each event. The guard on an event is a condition such that when it is true, then it is OK to let the event happen. The guard depends on the dependencies that have been specified. The approach proposed in [33], considers:

- Initially determining the guards on each event.
- Arranging for the relevant information to flow from one event to another.
- Modifying the guards to acquire the information received from other events.

The model in [33] can be viewed as a conceptual approach. The approach does not address the challenge of how the required declarative specification created at the first place.

Interleaving Web services composition and execution, using software agents and delegation have been discussed in [21]. Interleaving stands for carrying out the composition and execution of Web services in parallel. Furthermore, An *agent-based multi-domain* has been proposed in [21]. Domains are spread across the network and administrators manage them. Two types of domains exist: *"user-domain"* and *"provider-domain"*. A *"domain"* has been defined as a *"cluster"* of computing hosts on top of which portal of services and software agents are deployed. Two types of agents have been considered in the *"agentification"* process [21] of an environment of Web services: *"user-agent"* which acts on behalf of users, and *"provider-agent"*, which acts on behalf of providers who manage the Web services constituting the basic components of composite services. The work mainly focuses on the three dimensions along which a composition process can be seen:

- What set of services to use for composition process.
- Which service provider's resources to use.
- Which instances of each service to use for a particular client session.

Furthermore, it has been mentioned that, for a composite service, the selection of its component services is based on two criteria: *execution cost* and *location* of computing hosts.

Although the work proposed in [21] is utilizing software agent to Web services composition, however the model mostly focuses on the *selection* of services involved in the composition rather than composition itself. The argument is that a composite service cannot handle a change that occurs while it is under execution. However, the model does not discuss how to define a dynamic model for composition itself. On other words, before dynamic selection of services, we need to know how dynamically the processes in composite service are changing, which contribute the need for new services.

There are some other researches on service composition using both the concept of agents and some descriptive languages such as DAML-S and Web Service Flow

Languages (WSFL). Most of the researches have been focused on using DAML-S and DAML+OIL to propose a semantically approach for service composition.

Composing Web services using techniques from *"Agent Factory"* has been discussed in [29] and [34]. DAML-S [41] standard has been used as a description language to reason about Web services. The main goal of the work is to show how the DAML-S description of Web services offers enough structure for automated configuration by Agent Factory.

Agent Factory [34] has been defined as a service for automated design of software agents. An agent factory consists of:

- *Design Center.* Responsible for the actual design process of agent specification based on given requirements.

- *Building Block Retrieval.* Responsible for retrieval of building blocks by querying, in which characteristics with respect to functionalities, behaviors and states are specified.

- *Assembly.* Operational code is assembled on the basis of an operational configuration specification.

The approach in [34] considers just the design level. The design process is composed of three sub-processes:

- *Design Process Coordination*, coordinates the design process itself by issuing information related to overall design strategies.

- *Requirement Qualification Set Manipulation*, manipulates sets of requirements.

- *Design Object Description Manipulation*, manipulates descriptions of design objects.

In the model proposed in [34] it has been tried to draw a map between Agent Factory Structure and DAML-S concepts. It has been claimed that Agent Factory and Web services both poss a *conceptual* and an *operational* description. DAML-S is used to specify conceptual building blocks and WSDL to express operational level details.

DAML-S contains:

- *Profile*, which describes what the service does.
- *Process*, presents the internal working of the service in terms of the internal process, their process model and the internal data-flow.
- *Grounding*, specifies the operational level details of the service by linking the conceptual level descriptions to the WSDL description of the service.

Furthermore, the following map has been defined:

- Components in Agent Factory $\mapsto$ Service (with Process, Profile, and Grounding) in DAML-S.
- Data Types in Agent Factory $\mapsto$ IO and External Ontology in DAML-S.
- Coordination patterns in Agent Factory $\mapsto$ Control constructs for composite process and preconditions in DAML-S.

A representation composition constraint for semantic Web services has been proposed in [8]. The work has proposed the *Agent Service Description Language (ASDL)* to describe the external behavior of the agent services. An ASDL specification describes the messages understood by a service along with an interaction protocol. Like WSDL, ASDL can be published and accessed through a registry. The imported services represent an agent role with its behavior described in its ASDL description. In other words, by using ASDL it is possible to define the behavioral characteristics of a Web service. A Web service that implements behavior to preserve its autonomy has been considered as *Web Agent Service* [8].

In fact ASDL is a behavioral extension of WSDL, which describes the constraints of service invocation to capture the external visible relationship between the operations. An ASDL contains states for all operations defined in WSDL with the name and operation attributes same as the operation name.

Moreover, the [8] has proposed the *Agent Service Composition Language (ASCL)* to describe the composition of new services from existing ones. An ASCL specification describes the interaction with other agent services and describes the logic of how a new service is composed from existing ones.

In our point of view, although both works in [8] and [34] have proposed new and interesting ideas, however, both of them are unable to give a dynamic composite service. For instance, in [8], in order to define a composite service; the designer *must* create the corresponding files for ASDL and ASCL in advance, which are expression languages for service composition. The [8] does not address how ASCL and ACDL are generated during run time. In fact their methods are suitable to present and invoke those pre-known services, which have been involved in a composition. In case of changing or need to an individual service, the models do not give any answer.

A model based on Petri Net [26] has been proposed in [27]. The goal of the approach is to enable markup and automated reasoning technology to describe, simulate, compose, test, and verify compositions of Web services. The model uses DAML-S ontology for describing the capabilities of Web services. Moreover, the semantic has been defined in terms of *first-order logical language* and, the service description has been encoded in Petri Net formalism and to provide decision procedures for Web service simulation, verification, and composition [27].

Moreover, the model in [27], has provided a set of computational analysis tools, based on Petri Net, which provide automating Web service tasks such as:

- *Simulation.* Simulates the evolution of a Web service under different conditions.
- *Validation.* Tests whether a Web service behaves as expected.
- *Verification.* Establishes certain properties of a Web service.
- *Composition.* Generates a composition of Web services that achieves a specified goal.
- *Performance Analysis.* Evaluates the ability of a service to meet requirements with respect to throughout times, service levels, and resource utilization.

### 2.2.3 Agent-Based Approaches for Quality of Web Services

Current architecture of Web services technology allows a service requester to find services based on syntactical match and some criteria. However, current approaches do not provide any mechanism for selecting a suitable or good service. On other words, the challenge is, how to select and locate a service provider, who offers the best implementation of a particular service regarding cost.

### Reputation of Web Services

There have been some researches in proposing an agent-based approach to assist a service requester to choose the most suitable service. A conceptual model for Web service reputation has been proposed in [23]. In this model, there has been defined a *"Web Service Agent Proxy (WSAP)"* to access each service. A WSAP is an agent that acts as a proxy for clients of Web services. A client would have a WSAP for each service that it needs. In other words, when a client needs to bind to a service , it instantiates a WSAP, which consults the registry to get help in finding an appropriate service provider. To this extent, the registry records any feedback from the client to help finding better providers in future.

The WSAP model proposed in [23] can be considered as an interesting approach in rating mechanism. However, there are some questions and ambiguities in this model:

1. The model assumes that clients have already chosen the interface of the desired services, and just need to decide on a service implementation. This is a special case of service discovery, which has been reduced to *"service implementation discovery"*. There is still an open problem; in the case that how clients choose an interface rather than implementation.

2. The model addresses the rating mechanism on the client side by which each client has a WSAP for each Web service. However, there is no proposed model on UDDI registry side to show how the requests from WSAP are handled. On other words, the current technology of UDDI registry does not handle a rating mechanism by getting some information as minimum qualification and deciding on a good candidate to assist clients. The current paradigm of UDDI is just a

database of information, which is available through some query languages. However, in order to reason about qualification or reputation of a service provider there is no mechanism. In next chapter we will argue that besides needs for an agent based entity on the client side, there is a need to have an agent based component in UDDI registry as well, to carry out the clients requests.

A revised version of the WSAP model presented in [23] can be found in [22]. In the revised version, besides the *"Web Service Agent Proxy ( WSAP)"*, a third party named as *"Reputation and Endorsement System (RES)"* has been considered to facilitate the tasks of WSAP. The RES saves information regarding reputation. Moreover, the functionalities of WSAP have been changed a little. The WSAP is configured by a client with information on the service it process. All service activities of the client occur via the WSAP. In fact, *WSAP can monitor the activities and usages of the service by the client and help in future usages.* A WSAP can also offer and obtain advices from other WSAPs. WSAP's collect information on the services they bind to and convey this information to the RES.

In the revised version [22], each WSAP has a knowledge base, which is updatable for future use. In this case, it seems that the clients *must* keep each WSAP specified for any Web services to use in the future. However, there is a fundamental question that:

*"Is this model implementable in the real world where there are a huge number of Web services and still they are growing?"*

In our point of view, keeping information regarding of reputation and endorsement of any Web services on the client side if it is not impossible, but indeed, it is a tough task that no client would like to do. The key fact about the Internet based business is that, deciding about choosing a service should be fast enough. We argue that the information about reputation and history of a service should be computed in advance. This information should be capable of being updatable and more important must be reliable.

In the next chapter, we propose a model by which the client can query the rating information about each service more efficient and trustable. Moreover, in our model there is no need to keep track of rating information of services on the client side.

## QoS in Web Services

The issues of Web service delivery and QoS control have been discussed in [49]. The model discusses the QoS in Web services by proposing software agents architecture named as *WISE*. The agent-based WISE has been viewed on both Web service side and client side:

- Agents on User Devices. An agent at a *mobile* device is delegated to deal with a Web service server. For each Web service that is requested by the client's device, the WSDL information of the service is kept at the mobile device. Also, some QoS information about the service such as response time, history, etc. has been *kept* in the client's device as well.

- Agents on Web Services Server. Agents reside on the Web services servers and manage the context of the customers in order to assure more effective services. This type of agent can take the form of multi-agent systems, which are delegated to deal with the Web service server. The delegation involves management and utilization of profiles and context of all customers requests to make sure that customers are satisfied. The QoS capability of a WISE server can be published in its UDDI registry, so that service requesters may get help to find a suitable service.

In our point of view, the model suffers from some basic problems:

1. Based on assumptions, by each request of Web service, not only the WSDL information of the service is moved to the mobile device, but also some QoS information about the service such as response time, history, etc. has been *kept* on the client's device as well. However, the mobile device usually has a very short *memory* and *capacity*. Therefore, saving large amount of information as mentioned above on the mobile device cannot be considered as an efficient method.

2. The issues of QoS for an individual entity must be prepared by a third party and no individual entity can offer some unreliable information in terms of its QoS to the customer. Therefore, adding some information regarding QoS for a Web service, by its owner, on the UDDI registry is not acceptable by service requesters.

3. In the case of agents on client side, the model again saves QoS information as private one, which is not usable by public. The feedbacks of all Web services should be available for any service requester.

### 2.2.4 Utilizing Software Agents in Web Services Technology

There are a few research papers on utilizing agents in Web services technology. Agents have been used to overcome some specific shortcomings of the Web services technology.

Using agent-based system for the personalization of Web services has been presented in [18]. A *delegated* agent has been used to interact with the Web services. The authors argue that, agents to interact with Web services, must be able to do the following tasks:

1. Discovering Web services by agent with respect to customers preferences.
2. Delegating agent, which is capable of adopting its behaviors regarding circumstances.
3. Describing Web services functionalities to the agent by Web services themselves.
4. Communicating with Web services independently.

The proposed model uses agents to implement delegation. The personalization component consists of: *User Profile, Conversation Manager, Context Managers, Rating Service,* and *Agents.* Therefore, a user interacts with Web services via this personalization component.

In [18] the personalization component has been used by an individual client and specially in rating services, it just keeps the rating information privately and not for public use. Another important problem with this model is how agents interact with Web services and who is responsible for creating agents.

The implementation issues about relations between software agents and Web services have been discussed in [20]. To have a software agent access an external Web service for its own use or to offer the service to other agents, a *mapping* between the Web service description and the *FIPA Service Description (SD)* should be done. As an example, for each interface of Web services, or an endpoint of WSDL link, a FIPA SD entry in *Directory Facilitator (DF)* should be created to register the services.

Web services, either by UDDI or by software agents, must advertise their services. In case of non-agent clients, the description is available through the WSDL's URL link in the UDDI. In agent-based clients, the description of the Web service must be available through the Directory Facilitator. There have been defined the following elements to advertise services:

- *"Web Service Agent"*. Agents offer a Web service to general clients.
- *"UDDI Registering Agent"*. Dynamically generates the descriptions of agent-based Web services and register them in the UDDI registry.
- *"Platform provided Service Description and translation services"*. Provide support for describing agent-based Web services according to FIPA standards.

Generally, the author in [20] has investigated some issues involved in adopting agents in a multi-agent system for offering Web services to non-agent clients.

### 2.2.5 Examples / Related Projects

In this section, we review a few most related projects.

#### NIIIP

The *National Industrial Information Infrastructure Protocols (NIIIP)* project is a consortium of 15 organizations and U.S. government by developing a framework to allow manufacturing companies to communicate with their partners and set up a VE. It is one of the best known projects in this area. The developed protocols are based on some technologies [36] such as: CORBA, SQL, STEP, WfMC, and the Internet. NIIIP addresses integrated virtual enterprise in which several companies and their partners can work as a temporary single enterprise. The project gives opportunity to new enterprises to participate in a VE and use the latest distributed information technologies. Figure 2.2 shows the simplified model of the NIIIP system architecture.

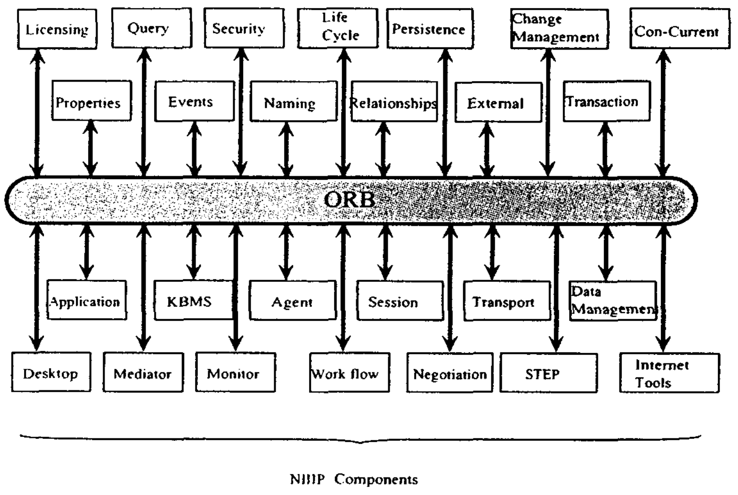


Figure 2.2: The NIIIP system architecture

The resources and services are provided by a set of servers, which are physically distributed, but are interconnected through an *Object Request Broker (ORB)*. Each service which is offered by a server is acceptable via *NCL, NIIIP Common Languages.* NCL allows each server to describe its services in a uniform method in terms of rules, methods, and their constraints. The services are acceptable via NIIIP's components. These components offer some requirements such as: session, negotiation, agent, data management, workflow, etc. All services and components communicate through the ORB. NIIIP is built upon a mediated architecture guided by a rule in knowledge based management system and utilizing intelligent agents to achieve integration and inter-operation.

### AgentCities

The *AgentCities Network* [40] is a global connection of agent platforms, services, and agents, which represents the dynamic service environment for the various AgentCities projects [37]. AgentCities [9] is a world wide initiative designed to help realizing the commercial and research potential of agent-based applications by constructing an open distributed network of platforms hosting diverse agents and services [47]. The main goal of the project is to enable the dynamic, intelligent and autonomous composition of services. The AgentCities network represents the first attempt to build an open, global and standard-based agent environment for commerce over the Internet. Communication and interaction in the AgentCities network are based on current available standards [38] such as: Foundation for Intelligent Physical Agents (FIPA) [42], W3C Consortium, DARPA, DAML, etc., as well as infrastructure services such as Web Services and ebXML. Three elements are involved in the AgentCities network [47]:

- *Agents.* A computational process that implements the autonomous communicating functionality of an application. They reside on an agent platform and are able to communicate to provide services to one another using the message exchange and communication features provided by each agent platform.
- *Services.* A series of actions carried out by a service provider. Services are registered in the AgentCities Network if they appear in the global Service Directory (SD).

- *Agent Platforms.* Software environments that support agents running in the AgentCities Network with access to support services. Each instance of an agent platform represents a single node in the Network. Agent platforms in the AgentCities network are registered in a simple centralized *Agent Platform Directory* (APD).

### RACING

*RACING* [39] [12], a Ukrainian project, focuses on applying agents to Web service composition for information retrieval. The goal of the RACING project is to provide *mediation* facilities for *content-driven* query processing such as query transformation and query decomposition. The main purpose of the RACING mediator is to produce *results* based on the *semantic matching.*

Agents have been used in almost every part of the project. For service composition purposes, the agents of the mediator and the agents that wrap their information resources participate in performing business processes of information retrieval by providing their Web services in a proper composition [12]. The architectural processes of the RACING project consist of:

1. *Service Requester Layer*, which consists of the *Service Requester Agent* capable of discovering UDDI's services.

2. *Middle Agent Layer.*

3. *Service Layer*, which consists of the primitive services and is able to generate *Service Providing Agent.*

The *Middle Agent Layer* is responsible for task decomposition, arrangement, credibility evaluation, monitoring, etc. The middle agent layer consists of a unit named as *Utility Agent*, which includes two agents: *Coordination Agent* and *Ontology Agent*.

User request processing in RACING is an *Ontology-Driven* process. The transformation methodology is based on incremental user profiling. The mapping of a user's keywords to the concepts of the domain ontology is built according to the transformation rules. These rules are based on the usage of the set of the semantic relationships.

The system consists of two FIPA-complaint agents [12]:

- *User Query Transformation Agent*, an agent, which has direct contact with the user and performs the query transformation.

- *Mediator Ontology Agent*, Mediator knowledge base management.

The *Mediator Ontology Agent* supplies the *User Query Transformation Agent* the contents of the user profile, RACING ontology, and the required portions of the domain mediator ontology.

## 2.3 Summary

In this chapter we reviewed three major concepts, the research literature, as well as some related projects which have been implemented in order to setup a virtual enterprise. We highlighted the existing problems in order to provide a clear picture of our objective in proposing a new model. In the next chapter we propose our multi-agent based model for Web services to use in a virtual enterprise environment.

# Chapter 3

# A Web Services / Agent Based Model for Inter-Enterprise Collaboration

In the previous chapter we introduced the concepts of virtual enterprise and the requirements to set up a collaborative environment. In this chapter we propose a multi-agent based / Web services model for enterprise collaboration. We define some agent based components inside an individual enterprise such that their integration with Web services technology assists the enterprise to involve into a collaborative environment. To this extent, a conceptual goal based model, for the dynamic workflow purposes, has been defined. Furthermore, this chapter presents an agent based model for service registry, which enables searching Web services semantically.

## 3.1 Enterprise Collaboration: Challenges

E-commerce and the Internet force the globe to move away from their traditional isolated systems and join into a world wide level business. Today some limitations such as geographical distances or incompatibility of resources are not allowed to cease businesses. Enterprises to persist competition must involve at a universal trading level. Virtual enterprises potentially shorten delivery time and cost as well as fulfill market demands for new services.

Although virtual enterprise seems being a suitable solution for today's businesses, but setting up and establishing a typical VE is not straightforward. Today's collaboration amongst enterprises offers some challenges to deal with:

1. **Diversity of Platforms.** Enterprises would like to share their resources in order to decrease cost and time as well as to carry out the market demands. Resources, based on their natures, consist of a large variety of categories such as hardware, software, data, methods, human skills, etc. In almost any case, each enterprise has its own legacy systems, which manages the resources. Due to *"diversity of platforms"*, on which resources are sited; enterprises are not flexible to communicate, exchange, and involve in a diverse collaborative environment. Enterprise collaboration demands a flexible technology to handle all communication, transportation, and integration amongst heterogenous environments regardless of their platforms.

2. **Tightly Coupled Components.** In today's collaboration, components are fully dependent on each other, or technically, they are *"tightly coupled entities."* In other words, changing one component in an individual enterprise, results in modifying the other components inside another enterprise. Consequently, any kind of modification, in an optimistic case, results in both cost and time. However, in a worsen case; it results in any unpredictable outcome such as inconsistency, incompatibility, reliability, and failure.

   The partners that are involved in any coalition *"must"* be free to change their data and environment, or generally legacy systems, without causing any problem for the VE. The partners involved in a VE, should not get some extra burdens to change their systems in terms of compatibility. Today's E-business offers a challenge to create a collaborative environment, at which the components are *"loosely coupled"*.

3. **Service Locating.** Companies, for collaboration purposes, need to discover and locate the desired resources or services, which are offered by some other enterprises. Traditional methods, such as advertising, in finding a company with specific sharable resources or services are too costly and time consuming. However, in a collaborative environment locating a suitable business parter must be fast enough. Moreover, in most cases, it is difficult to find the best *"qualified"* partners in terms of the requirements. In fact, the challenge is: how to trust, or rate a service provider and measure its services. Generally, one of the most important challenges in establishing a VE is finding suitable enterprises, which best fit the requirements of a typical virtual enterprise.

4. **Static Process Definition.** In order to achieve a desired goal, integration of partners in a VE needs to define a set of sequenced processes, and assign them to the partners for carrying out, and contribute to acquire the common goal. A workflow is a collection of ordered business processes towards achieving a final goal. Each process consists of some sub-processes or tasks, which are performed by some partners. The *"pre-defined"* and *"pre-ordered"* processes in designing a workflow conducts to a static nature, which has been created at design time rather than runtime. In fact, the static nature of a workflow defined in a VE prevents the participants to acquire some unexpected opportunities. These

surprising chances could enhance and affect the overall result. Basically, highly growing of enterprises and their enhancements in services, create a challenge to dynamic defining of processes and their orders in a virtual enterprise.

5. **Module Redundancy.** Today's software technologies for establishing a virtual enterprise suffers from redundancy in developing objects, which have been implemented before. Software objects are not reusable throughout a heterogenous environment such as the Internet. Some technologies, such as CORBA, are addressing reusability as a limited approach. But they have some limitation to be used through heterogenous platforms. Reusability and modularity of software components help developers implementing faster and more reliable, which save both time and cost. So, the challenge is, in case of needs to an object, which already exits, how an enterprise can use it without developing from the sketch.

## 3.2 Integration of Web Services and Software Agents

Web services technology, a new paradigm in distributed computing, by offering many great features, seems providing some solutions to address the VE's challenges and meet the requirements for creating a virtual enterprise. Regarding the considered challenges in a virtual enterprise in previous section, the Web services technology provides some values to address those challenges.

1. **Platform Independency.** Web services technology provides communication among participants at the application level, by which enterprises are able to develop or maintain their legacy systems developed by any language or platform

and share resources through heterogenous environment. The platform independency feature of Web services technology is considered as a main revolutionized value, which addresses the diversity of platforms challenge in a virtual enterprise.

2. **Loosely Coupled Components.** Enterprises, by using Web services, are able to create some independent components, which in spite of getting involved in any virtual collaboration, are free to be modified, replaced, and even removed without affecting any other component in the collaboration. In fact, Web services technology provides an environment, at which enterprises freely take full control of their legacy systems regardless of existing components of the other participants. This feature of Web services technology addresses the tightly coupled component challenge in a virtual enterprise.

3. **Service Registry.** Web services technology by introducing the UDDI facility, has given some great opportunities to enterprises for choosing their suitable business partners in a more efficient way. Enterprises by registering through the UDDI registry, advertise their offered services more effectively, which saves both time and cost. Consequently, a service requester, by searching the UDDI database, discovers the desired enterprises, who are offering the requested services. The three general discovery styles of information known as: *white, yellow*, and *green* pages, provide all required information to make a decision about choosing a business partner. The UDDI facility, enhances both the creation and evolution levels of a virtual enterprise's life cycle, at which the coordinator, in order to locate or replace an existing participants, refers to the UDDI and discovers another service provider as a new business partner. The UDDI facility of

Web services technology addresses the challenge of service locating in a virtual enterprise.

4. **Dynamic Definition of Process.** The feasibility of defining a *"service composition"* in Web services paradigm is considered as a great feature such that by stacking some value added services, a new service, or a new process model will be defined. The composite service enables a VE to define, sequence, and assigns the subprocess to the eligible service providers. Also, a VE is able to change the composite service at runtime, which contributes to define a dynamic workflow. This feature addresses the challenge of static definition process in a VE.

5. **System Modularity.**The defined methods or data inside a Web service can be invoked and used in any application regardless of platform. The *"modularity"* of components in Web services makes them *"reusable"* and accessible through the Internet. As a result, in case of developing new software, there is no need for implementation from the sketch. Application developers, simply by just invoking and using the methods and data, which have been defined in Web services, create their products faster and more reliable. Thus, the participants involved in a VE, in case of needs to an object, which already exists, adopt it without any need for developing from the beginning.

In spite of providing glorious features in distributed computing, the Web services technology does not address a complete solution for enterprise collaboration. The problems can be viewed through a wide variety of the Web services technology.

We believe the *"Distributed Artificial Intelligence (DAI)"* can be considered as a complementary paradigm to overcome some shortcomings of Web services for adoption into a VE. DAI has a powerful potential for offering some intelligent properties to a *"dumb"* Web service. We argue that the combination of Web services and software agents provides more flexibilities and features. Adopting the highlighted properties of software agents such as: autonomy, adaptability, sociability, and pro-activeness in the Web service technology, contributes to proposing the *"intelligent Web services"*.

Adopting intelligent software agents extends the Web services technology in some important issues:

- A Web service is just a self-describing software component such that, it has knowledge just about itself. A Web service does not have any idea about around environment, users, software components, and generally outside world. In a collaborative environment such as virtual enterprise, participants should be smart enough to locate other enterprises, their resources, and offered services. In contrast, software agents have some meta-data, by which they are capable of reasoning, learning, and interacting with outside world, or in fact, other enterprises, which are involved into a VE.

- Web services are discoverable by the XML-based UDDI standard. Current standard of the UDDI is just able to recognize terms *"syntactically"*, which are matched with the definition of services. If a software component tries to discover a Web service with different ontology, the result of discovery would be unpredictable. The main challenge in service discovery is how to find services, which are *"semantically"* the same as clients' desires. Moreover, current technology of Web services does not address any solution in finding the most suitable business partner. There is no ranking mechanism or knowledge to assist service requesters for choosing the best services. Software agents, by achieving the service requesters' preferences, are able to reason and assist the service requesters towards choosing the most reliable and suitable services. Also, by adopting some techniques such as knowledge representation, an agent is able to locate the desired services semantically.

- Web services are known as quite and *"passive"* components. They never start communication. In a competitive business world, service providers should introduce their services *"pro-actively"* in advance rather than asking them to present. In a typical virtual enterprise, where there exist enormous business enterprises, the pro-activeness is a key success. In contrast, agents are pro-active entities that like to find other components and start communication. Agents are capable of being aware of any changes in their domain to acquire information from the outside world and adopt their internal knowledge.

- In order to use a service, a requester by analyzing the content of Web Service Description Languages (WSDL) finds out: how to *"invoke"* the desired methods, how to submit the arguments, and how to receive the return values. This kind of *"calling"* a Web service cannot address meeting all the requirements of a service requester. Agents provide a mechanism such that a service requester can *"request"* a service rather than *"invoke"* it. In fact, agents by achieving requests from service requesters, try to map them to the existing methods. Therefore, users do not need to know any thing about the internal structure of the requested services. Adding this feature to a collaborative environment enhances the communication among participants. In fact, participants to receive a service, simply ask for service rather than looking for syntax of calling and realizing how to use it.

The combination of intelligent agents and Web services technology enhances the ability of realizing and handling requests. In fact, service oriented paradigm and agent-oriented technology together are more powerful to provide a better solution for enterprise collaboration. We believe, the concept of VE, the functionality of Web services and the intelligence property of an agent together offer a more effective electronic collaboration of enterprises over the Internet.

## 3.3 A Web Services / Agent Based Model

In this section, we propose a *"service oriented / multi-agent based"* model to setup a virtual enterprise. We have already clarified the weaknesses of the Web services technology towards creating a collaborative environment and the strengths of the software agent technology as a complement of the Web services technology. Our approach will be based on adopting the agent paradigm to enhance the functionalities of some weak areas of the Web services. We focus on integrating Web services and software agents inside the internal structure of a typical enterprise as well as adopting software agents inside the UDDI registry. On the enterprise side, besides considering Web services as the main components, we propose a goal based model to provide feasibility of defining a dynamic workflow and also we introduce some other agent based components for coordination, and monitoring purposes. On the UDDI side, we introduce some agent based components to carry out and assist service requesters towards choosing the most suitable and trustable service provider.

### 3.3.1 System Architecture

The proposed architecture is depicted in Figure 3.1.

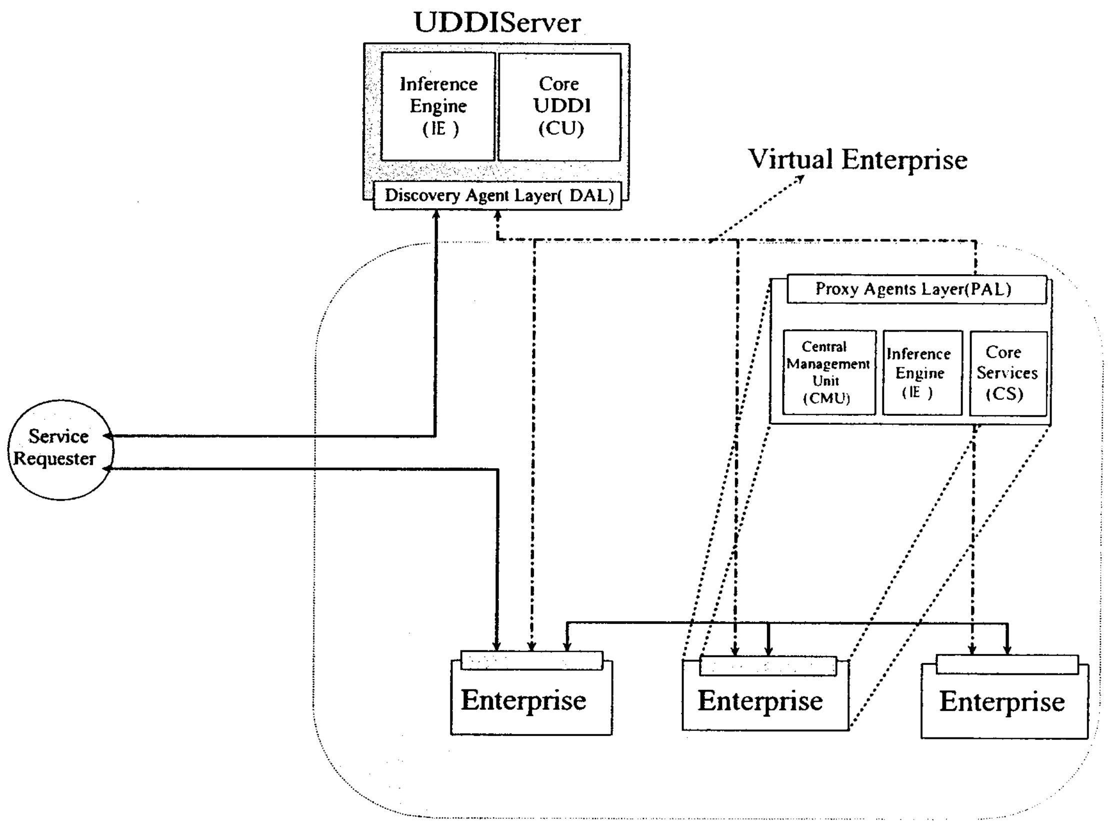


Figure 3.1: A Web services/multi-agents based model for enterprise collaboration

In this model, we have defined some agent based units both inside an enterprise and the UDDI registry server. On the enterprise side, A *"Proxy Agents Layer (PAL)"* is defined, which is sited on the top of other software units. PAL acts as an interface between the enterprise and outside world. A corresponding interface is defined for the UDDI server as well. The interface is named as *"Discovery Agent Layer (DAL)"*, which is sited on the top of the UDDI server. In fact, in the proposed model, we treat each Web service or UDDI server as an agent based component, which may act as a service provider, or coordinator.

As it is depicted in Figure 3.1, a service requester, who needs some sort of services, either primitive or composite, *"asks"* the UDDI server through the *"Discovery Agent Layer (DAL)"* to assist in locating a *"suitable"* and *"reliable"* service provider. Consequently, DAL by doing some mechanisms (we discuss about these mechanisms later in this chapter) along with some other components of UDDI, presents the *"recommended"* service providers to the client. Furthermore, by choosing the recommended service provider, which has been *suggested* by UDDI, the client *"requests"* the desired services from the chosen enterprise through its *"Proxy Agents Layer (PAL)"*. In our model, a client can be a service requester, provider, or both. A client is defined as both a service provider and requester if it is responsible for fulfilling some assigned processes and is unable to carry out all of them independently. In this case, the client needs to find a suitable partner to delegate some of those processes and create a virtual enterprise.

In following we describe each component of the architecture in detail.

### 3.3.2 Enterprise Model

We have defined four internal units for any individual enterprise as following:

- *Central Management Unit (CMU).*
- *Proxy Agent Layer (PAL).*
- *Inference Engine (IE).*

- *Core Services (CS).*

In the model, all interactions between an enterprise and outside world are carried out by Proxy Agent Layer (PAL). PAL is responsible for any incoming or outgoing messages and generally all interaction. The *"Core Services (CS)"* unit consists of two components as: Web services and legacy database. The definition and functionalities of the Core Services (CS) are the same as current paradigm of Web services including the enterprise legacy database. The other two units named as: *"Central Management Unit (CMU)"* and *"Inference Engine (IE)"* are responsible for defining or managing workflows and knowledge representation, respectively. We will describe each of them in detail in this chapter. The detailed proposed model for an enterprise likely to involve into a virtual enterprise is depicted in 3.2.

In following we describe each defined unit and its functionalities in detail.

### Central Management Unit (CMU)

The *"Central Management Unit (CMU)"* can be considered as an *"organizer"* component, which is in direct contact with the Proxy Agent Layer (PAL). In order to setup a VE, this unit has a leadership role in defining, designing, and assigning activities defined in the life cycle of the VE. The CMU is responsible for:

- Accepting requests from PAL to design workflows.
- Informing PAL to locate suitable service providers and delegate the tasks of workflows to them.
- Assigning the tasks to some qualified enterprises.
- Coordinating the workflow amongst business partners.

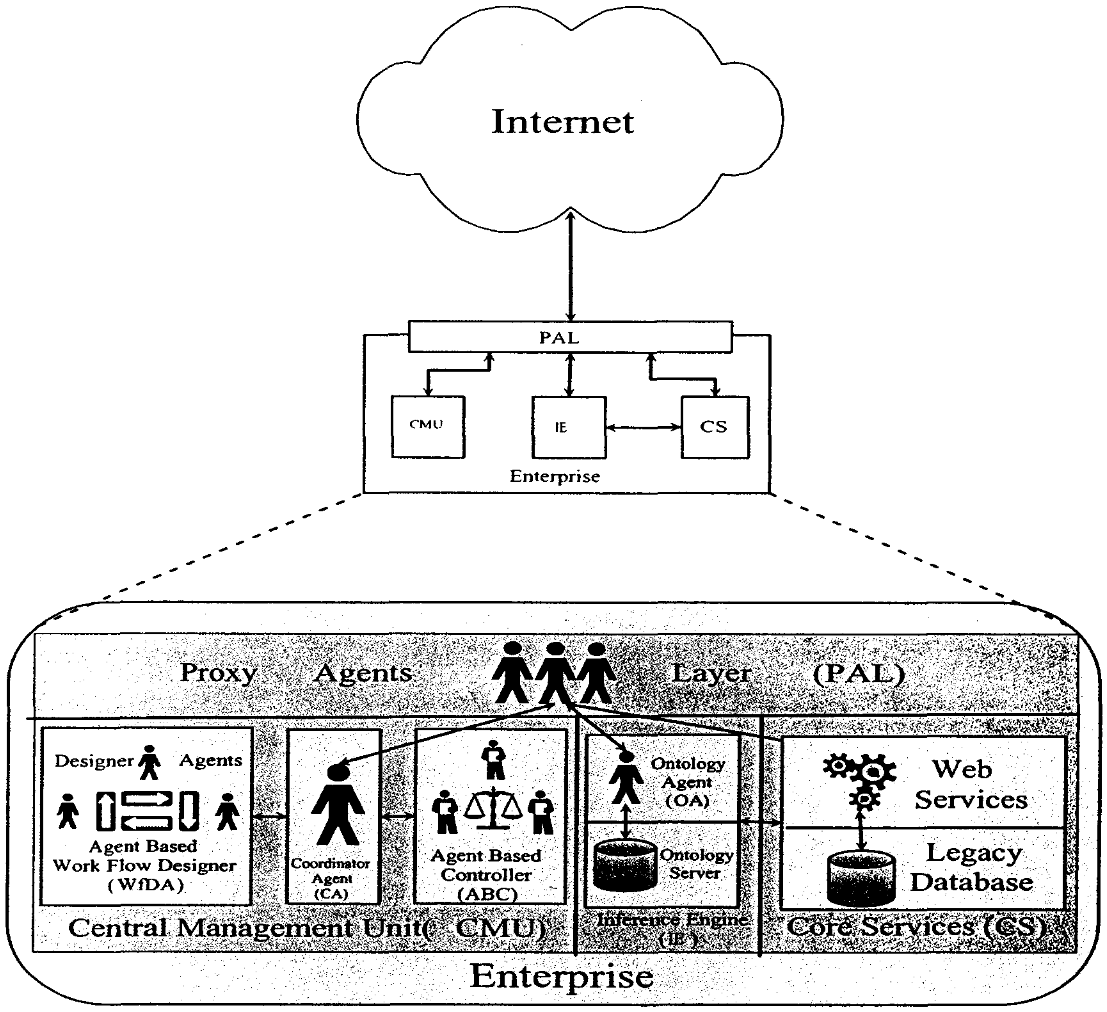


Figure 3.2: The internal architecture for an enterprise

- Monitoring the progress and status of the delegated tasks.

CMU consists of three subunits as following:

**Workflow Designer Agents (WfDA).** Generally, any kind of client, either human or agent, is interested in achieving goals regarding affordable cost and time rather than how to do so. This kind of view, leads us to consider designing *"Goal-Based"* workflows. Goal-based reasoning is one of the strong features of intelligent software agents. By introducing the goal-based workflows, we look at each workflow as a goal or set of goals, which are supposed to be gained. In fact, we are not interested in defining subgoals at design time as traditional methods; rather we let agents, themselves, designing the best and suitable subgoals based on some templates or knowledge. Eventually, we are just interested in submitting the ultimate goals to agents rather than *"how"* to achieve them.

The Workflow Designer Agents (WfDA) unit, is responsible for almost all activities defined at creation and evaluation level of a typical virtual enterprise's life cycle. Consequently, a WfDA will be in charge of: designing workflows, partner searching, agreement, delegating tasks, and monitoring the status of the delegated tasks. However, in our model, not only WfDA is in charge of managing all activities defined at the creation and evaluation level of a virtual enterprise. In fact, those activities, either at creation or evaluation level, are distributed amongst some other components of the central management unit (CMU) as well. We discuss about each of them later in this chapter.

The most important activity of a WfDA is designing a *"cross-enterprise workflow specification"* at *runtime*, by which it is possible to create a dynamic, not only in specification but also in assigning tasks to partners, with respect to their qualifications.

Apparently, the starting point in a WfDA for an individual enterprise will be the issue of achieving a requested goal, which has been asked by client, who would like to share some resources or uses some services and creates a virtual enterprise. The underlined goal can be viewed as a requested service or product.

Let us considering a simple scenario. At the beginning the enterprise, through the Proxy Agents Layer or PAL, receives a request from a service requester for a service. PAL tries to map the incoming request to an existing available service at the existing Web services. This process takes place through direct connection between PAL and the *"Core Services (CS)"*. In an affirmative case, in which there exists an available service that can handle the request, the routine will be an ordinary request and result will be sent back to the requester, without any dealing with WfDA. Existence of a requested service results in a straightforward joining into a virtual enterprise, which has been created by service requester and without any effort to look for another service provider, or business partner.

However, in a negative case, in which PAL could have not mapped the request into an available existing service in CS, PAL acts as a ***"broker"*** to either find and assign the request to a service provider. Consequently, PAL has to do the following steps:

- PAL contacts with the UDDI server to find out if there exists any registered service provider, which can carry out the request, or goal. In affirmative case, in which there exists a registered service provider in the UDDI entry, PAL with respect to the *"user preferences"* such as: cost and time, which have been defined in request time by service requester, delegates the underlined request to the new service provider as the business partner involved in a new VE created by the enterprise itself. User preference, simply, is a vector of parameters or attributes, which is defined and set by the service requester. The attributes of the vector show the minimum acceptance and qualification of the requested service. To this extent, in order to monitor the progress of the delegated task, PAL informs CMU to create a *"controller agent"* for controlling the status of the delegated task. More precisely, the *"Coordinator Agent"*, or CA of CMU, by receiving the request from PAL for creating a controller agent, informs the *"Agent Based Controller"*, or ABC, to create the agent. We will discuss about the Coordinator Agent (CA) and the Agent Based Controller (ABC) in detail later in this chapter. In a negative case, in which PAL could have not found any eligible service provider at the UDDI, which is able to handle the request with respect to the *"user preferences"*, PAL follows the next step:

- PAL submits the request, or the goal, with its preferences to the *Central Management Unit* (CMU). The *Coordinator Agent (CA)* inside CMU sends the goal to the WfDA. The goal based WfDA decomposes the main goal into some other subgoals, as *"goal decomposition"*. The decomposition is based on some *"knowledge"* or *"templates."*

It is important to note that, handling the user requests and joining into a virtual enterprise, completely depends on either finding an existing service or decomposing the main goal into some existing subgoals, which can be handled by some existing services. It is clear that, failing to decompose a goal results in failing to satisfy a customer. Therefore, the decomposition methods and probably the available templates play very important roles in handling requests. The same scenario, which has been done for the main goal, *must* be repeated for each subgoal as well. The task decomposition and repeating the scenario will continue until availability of achieving all subgoals and eventually, the main goal.

We have to emphasize the role of the *"Coordinator Agent (CA)"* through creating the controller agents at the Agent Based Controller unit, or ABC, and assigning them the controlling of activities for delegated tasks. There must be a *"one-to-one"* corresponding between the created controller agents and decomposed subgoals, to check the status of each of them and consequently, the progress of workflows.

To prevent loop or endless decomposition, it would be reasonable to define a *"decomposition level degree"* to prohibit the WfDA of decomposing more than a certain stage. We define the first goal requested by client as *"main goal"*, the generated subgoals as *"intermediate goals"* and the latest goals with respect to the decomposition level degree as *"basic goals"* or *"atomic goals"*.

We believe that the traditional definition of workflow is not suitable for our model. Due to static nature of traditional definition and in contrast goal based nature of our model, we argue the needs to define a new goal-based definition for workflow.

**Definition 3.3.1. Traditional Workflow Process.**

*A tuple $W = (T, E_W, R_W)$ in which:*

1. *$T = \{t_i | 1 \leq i \leq n\}$, is a set of tasks to fulfill the process.*
2. *$E_W = \{ECA_{t_i} | 1 \leq i \leq n\}$, is a set of* Event-Condition-Action (ECA) *rules, which happen during the execution of each task.*
3. *$R_W$ is the result of workflow process, which can be either* success *or* failure. $\square$

The static nature of this definition implies that each process or task should be *predefined* and its execution will trigger some *pre-known Event-Condition-Action* rules.

Based on our goal-based idea, we propose a new definition for workflow, which has dynamic behavior, and changes with respect to circumstances. First we consider our proposed definition of goal.

**Definition 3.3.2.** ***Goal*** $(G, Q_G)$.

*A goal is a tuple $(G, Q_G)$ such that:*

1. *$G$ is the desired goal, which could be a set of subgoals as: $G = \{g_i | 1 \leqq i \leqq k\}$, in which $k$ is the number of subgoals.*
2. *$Q_G$ is the desired preferences in achieving the goal $G$, which is a set of acceptable qualification parameters as: $Q_G = \{q_n^G | 1 \leqq n \leqq m\}$, in which $m$ is the number of preferences in accepting the goal $G$.* $\square$

Now with respect to our definition of goal, we propose a goal-based workflow process (GBWf).

**Definition 3.3.3.** ***Goal-Based Workflow Process (GBWf).***

*A GBWf is defined as a tuple $W \triangleq (\{G_i\}_{i=1}^{p}, \{R_i\}_{i=1}^{p})$ such that:*

1. *p is the number of (sub)goals involved in a workflow.*
2. *$\{G_i\}_{i=1}^{p}$, the set of (sub)goals to define tasks in a workflow.*
3. *$\{R_i\}_{i=1}^{p} \in \{Success,\ Failure\}$, is the set of results of fulfilling of (sub)goals.*
4. *$o(G_i) \prec o(G_{i+1})$, is the order of achieving goals, which means ith goal, $G_i$, will happen before the $(i+1)$th goal $G_{i+1}$.* $\square$

We believe that this kind of goal based definition for workflows, or GBWf can best meet the requirements of the proposed model. The definition covers dynamic issues, since each goal is decomposed into some subgoals at runtime. The result of each goal or subgoal, with respect to its dynamic acceptable qualifications, or $Q_G$, can be success or failure. Usually, the content of $Q_G$ is considered as *"User Preferences"* for achieving goal. The preferences are some issues such as *time, or cost.*

Moreover, order of happening of goals has been considered as well. As a conceptual point of view, the order of achieving goals can be viewed as a *"rooted tree"* in graph theory, by which the root node is the same as main goal. Consequently, each node of tree is as a goal and each edge as the process of achieving the goal. Each node, or goal, is as a parent of some other nodes, or subgoals, and vice versa. Therefore, a Web service receives a request for fulfilling a goal such as $(G, Q_G)$, in which $G$ is as the requested service and $Q_G$ is the *"User Preferences"*. This tuple is received by PAL and PAL process on it as we described.

The goal decomposition, or task decomposition, is a complex topic in AI and we do not cover its techniques in this thesis.

**Coordinator Agent (CA).** All interaction between CMU and PAL are carried out by the *Coordinator Agent, CA*. The Coordinator Agent (CA) is responsible for coordinating and managing the processes of a workflow, which have been designed at WfDA.

The following notations have been used for describing the flow of the processes:

- $X \rightsquigarrow Y$ means $X$ informs $Y$.
- $X \xrightarrow{T} Y$ means $X$ sends the $T$ to $Y$.

The main significant duties of CA are as following:

1. Communicating with PAL:
   - $PAL \xrightarrow{(G,Q_G)} CA$. Sending the user request, which is the goal $G$ and its preferences in order to analyze towards decomposition.
   - $CA \xrightarrow{\{(g_i,q_{g_i})\}} PAL \rightsquigarrow UDDI$. Sending the set of decomposed subgoals and their preferences to PAL in order to, either match with the existing services in CS or contact with UDDI and search for an eligible service provider to carry out a specific (sub)goal.
   - $CA \rightsquigarrow PAL \rightsquigarrow EnterpriseHost$. Informing PAL to contact with the business partner's host, which is responsible for carrying out the delegated task to acquire the status of the delegated job for monitoring purposes.

- $CA \rightsquigarrow PAL \rightsquigarrow UDDI$. Informing PAL to contact with the UDDI server to report the output and qualification of an enterprise. This information will be saved on the UDDI server as history of an individual enterprise and is useful for ranking and future reference.

2. Interacting with WfDA:

   - $CA \xrightarrow{(G,Q_G)} WfDA$. Submitting the main goal, *"composed goal"* to the WfDA, for decomposition purposes.
   - $WfDA \xrightarrow{\{(g_i,q_{g_i})\}} CA \rightsquigarrow PAL$. Achieving the set of subgoals and their preferences, which are result of decomposition purposes, to deliver to PAL and find out how to handle them, (either by available existing services or by contacting with UDDI server).

3. Coordinating all involved enterprise partners to fulfill the workflow defined in the WfDA.

4. Communicating with the *"Agent Based Controller, (ABC), "* as:

   - Informing ABC to create a corresponding controller agent and compare the plan and the real progress. By delegating each task to an enterprise, CA informs PAL to contact with the enterprise host server to ask about the status of the process. Consequently, CA informs ABC to create a *"Controller Agent"*. The Controller Agent created by ABC is responsible for keeping track of plan of the delegated task and comparing them with the feedback information achieved via the communication between PAL and the enterprise's host. The processes is as following:

     (a) $CA \rightsquigarrow ABC$, creating Controller Agent.

(b) $CA \rightsquigarrow PAL \rightsquigarrow EnterpriseHost$, to contact and find out the status of the delegated task.

(c) $EnterpriseHost \rightsquigarrow PAL \rightsquigarrow CA \rightsquigarrow ABC$, the enterprise sends back the result of asking for the status of the delegated task.

(d) $ABC$ checks if Result $\overset{?}{\equiv}$ Plan.

- Achieving information from ABC, regarding the progress of the delegated tasks, to make a decision such as: replacing the business partners, or informing PAL to send a reporting message to the UDDI server regarding the qualification and performance of the enterprise and keeping track of information.

  (a) $ABC \rightsquigarrow CA$, informing the result of comparison.

  (b) $CA$ makes a suitable decision.

  (c) $CA \rightsquigarrow PAL \rightsquigarrow UDDI$, informing the output of the enterprise for ranking purposes and save in the UDDI registry.

5. Decision making in case of realizing the weakness of a business partner by receiving the *"Progress Report"* from $ABC$.

6. Synchronizing, or rollback, the whole transactions of workflow in case of *"drawback"*. In the Web based technologies the failure of fulfilling is highly possible. In a transactional workflow, the whole processes defined in the transaction either must be carried out, or rolled back.

WfDA designs workflows and submits the goals and the requirements to CA. CA in order to handle the goals, communicates with PAL to find out either the existing available services can carry out the goals or looking for some other eligible partners registered in the UDDI entry. By assigning each goal or task to a business partner, CA creates a controller agent in ABC unit, to keep track of the progress of the delegated tasks.

**Agent Based Controller (ABC)** The evolution stage of a VE has significant impact on the overall performances and the ultimate result. By monitoring the output of each partner, and comparing its result with plan, and probably replacing a weak partner with the better one, the overall quality of workflow can be enhanced. In our model, CA after assigning each task to an individual business partner, informs ABC to create a corresponding *"controller agent"* to keep the plan and monitor the progress of the underlined process, which is carried out by the partner. Accordingly, CA informs PAL to contact with the enterprise, which is responsible for doing the delegated task, and asks to report the status of the job. By getting a feedback from the enterprise, the controller agent that is sited at ABC unit, compares the result of the enterprise's activities with plan and informs CA to make a decision. Furthermore, CA informs PAL to contact with the UDDI server, to update the partner's profile for rating, reputation, and history purposes.

### Inference Engine (IE)

We argue that, besides needs to an ontology unit at the UDDI server to discover a desired service, (we will discuss about agent based UDDI server in the next section), it is essential to have an *Inference Engine, IE,* on each enterprise as well, for realizing the meaning of the incoming requests.

In current technology of Web services, clients by acquiring some information from

WSDL, start to exchange data with the underlined Web service. This kind of syntactically *"invoking"* a service cannot cover the semantically *"requesting"*. Therefore, an enterprise needs an intelligent component such as an *"Ontology Agent (OA)"* to realize and discover the incoming messages to the enterprise and try *to map them to an existing services*, or methods. Moreover, any agent in order to be able to reason about context needs a knowledge base engine or *Ontology Server*. The structure of IE in an enterprise system architecture and the IE, which will be proposed at the UDDI server in the next section is similar. Thus, The main responsibility of IE is to analyze the incoming messages with respect to semantics rather than syntactics. We will discuss about IE in detail in the next section.

### Proxy Agents Layer (PAL)

Current technology of Web services is known as *service-based* rather than *role-based.* As we mentioned, the ongoing paradigm over the Internet is toward providing an environment, in which components can be considered based on their roles and qualifications. The *Proxy Agents Layer (PAL)* can be considered as a complement component for Web services technology to change a passive enterprise to a proactive and intelligent entity, which offers some facilities as services and is capable of involving in transactions pro-actively. All communications between an enterprise and outside world will be through PAL. Moreover, PAL is responsible for exchanging data among internal components (CMU, IE, CS) of the enterprise as well. From another point of view, PAL can be considered as a *wrapper*, by which the functionalities and complexities of the internal structure of an enterprise and their descriptions are *encapsulated* from outside visions.

By adding the Proxy Agents Layer (PAL) to an enterprise, and creating an agent oriented Web services for enterprises, the establishing of a VE will be more efficient. As a very simple scheme, suppose an enterprise, which has received an order from a user. If the enterprise is able to handle the request by its existing services then the service requester is responsible for setting up and coordinates the VE. However, if the enterprise needs to another service provider handling the request, it creates a new VE. The main enterprise plays the role of coordinator aw well, and we call it *"root"*. Root is responsible for carrying out user requests. Consequently, root is in charge of locating and coordinating business partners, and monitoring their activities. Moreover, the partners, which are agent oriented Web services as well, are free to decide about their own internal transactions processes. On other words, the internal behavior of each participant is encapsulated from the others view.

It is important to note that, PAL *must* be knowledgeable about the current set of Web service standards as: SOAP, WSDL, UDDI. Otherwise, it cannot realize any incoming or outgoing messages. As a matter of fact, it should be able to parse WSDL to realize the characteristics of Web services. Also, realizing the SOAP messages and the UDDI standards also are a *mandatory* requirement for PAL.

The main responsibilities of PAL are:

1. Routing all incoming or outgoing messages to suitable software components either inside the enterprise or outside world. The interaction can be:

    - Receiving a request for an available existing services. Simply PAL sends the request to the Core Services (CS) and CS sends back the result to deliver to the user.

    - Communicating with the Inference Engine (IE) in case of existence of any ambiguity about the meaning of the used terminology.

- Communicating with the Control Management Unit (CMU), to decompose the requested service.

2. Exchanging information between the internal components of the enterprise such as between IE and CS or between CMU and CS.

3. Encapsulating and wrapping the functionalities of Web services.

4. Communicating with other enterprises (negotiation, coordination, etc.).

5. Contacting with other partners to check the status of the delegated tasks.

6. Contacting with the UDDI server, either looking for a partner or reporting the quality of services.

### 3.3.3 UDDI Servers Model

The current technology of registering and discovering of services in the UDDI server has been developed for human users. If we imagine finding a suitable service from enormous number of entries of services with different properties provided by different enterprises, then it would be clear that locating a suitable service, if is not impossible, clearly is difficult. As a result, there exist needs to an intelligent assistant component such an agent to assist service requester in finding a suitable business partner. In Web services paradigm, the most interesting application of an intelligent agent can be found in discovery of services. In following we propose an agent oriented architecture for the UDDI server.

We have defined three internal units for a typical UDDI server as following:

- *Discovery Agent Layer (DAL).*

- *Inference Engine (IE).*

- *Core UDDI (CU).*

All interactions between a UDDI server and outside world are carried out by *Discovery Agent Layer (DAL)*. DAL is in charge of any incoming and outgoing messages. In fact, DAL acts like an interface to accept the requests from outside world and analyze them by cooperation with the other components. The proposed architecture for a UDDI server is depicted in Figure 3.3.

As it is clear from Figure 3.3, the existing technology of the UDDI server is just as the *"UDDI Database"*, or a pile of *hardly* discoverable data, which is not able to assist requesters straightforwardly. The *Core UDDI (CU),* consists of some information about service providers, which are included in an ordinary UDDI database server, as well as keeps some other information of service providers in terms of the *"quality of services"*. CU also consists of an intelligent component named as *"Rating Agent"*, RA, for QoS purposes. The *Inference Engine (IE)*, is in charge of realizing and interpreting the services requesters' messages semantically rather than syntactically. IE consists of an *"Ontology Agent"*, which is responsible for reasoning about the meaning of terms. Moreover, an ontology server is defined as knowledge representing for OA.

As a simple scenario, suppose the UDDI server receives a request to locate a suitable service with some specific qualifications. DAL, as an interface, receives the request and tries to find the suitable services, which match the request. Then:

1. In case of finding some suitable services, DAL submits the qualification requirements, or preferences submitted by the service requester to the Rating

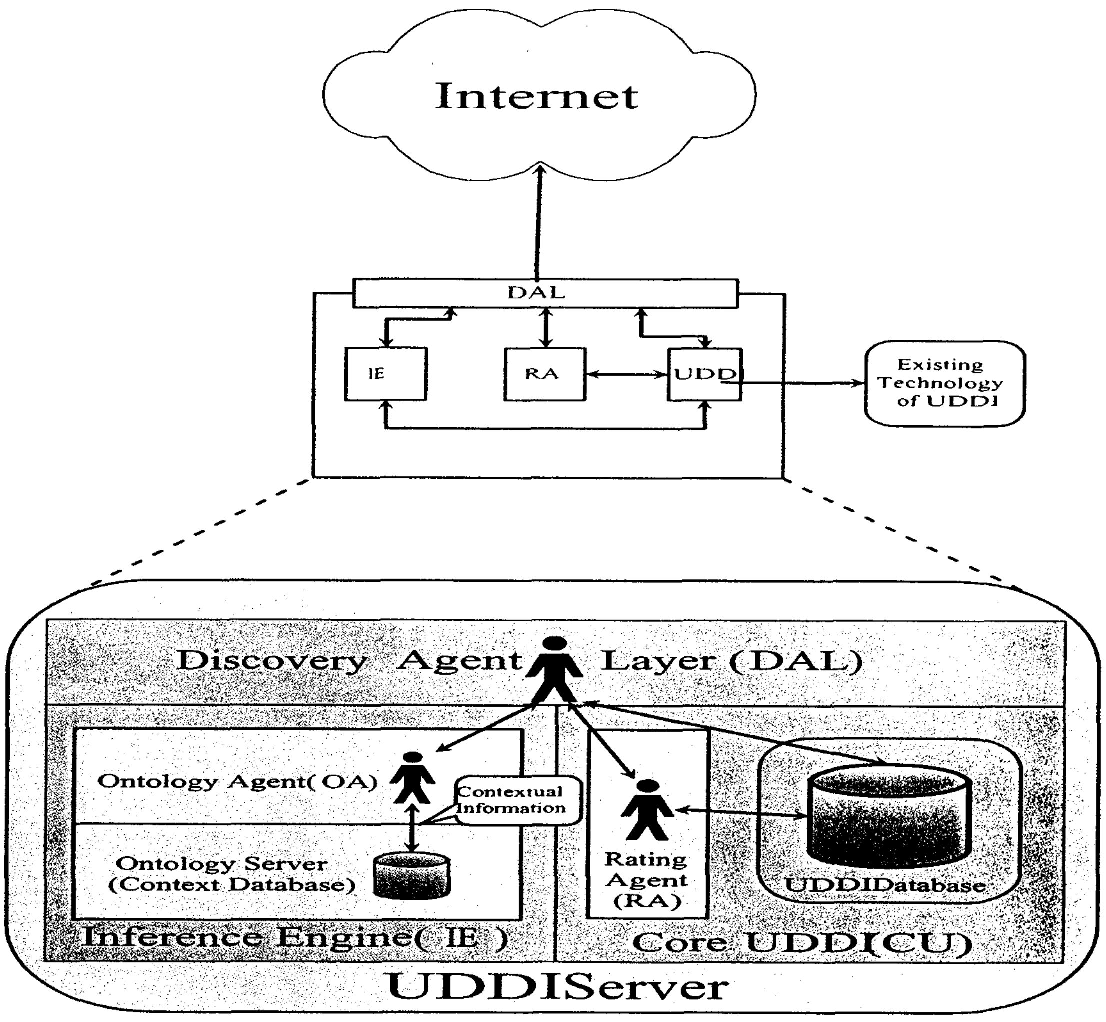


Figure 3.3: An agent-oriented UDDI architecture

Agent, or RA. The qualification requirement can be a *"set of parameters"*, or a *"vector"*, which consists of some *"attributes"*. RA by reasoning *"suggests"*, or *"recommends"* the most suitable candidate to DAL. Finally DAL sends back the identification of the suggested services or service providers with respect to their qualifications, to the requester.

2. In another case, in which DAL could have not matched any registered services with the request, DAL sends the terminologies of the request to IE to find out the meaning of the request. OA by analyzing and reasoning, and by using the ontology server, sends back the meaning of the terms to DAL. Furthermore, DAL repeats the same scenario to find suitable matches semantically.

In the following, we describe each component in detail.

**Discovery Agent Layer (DAL).**

The *Discovery Agent Layer (DAL)* acts like an interface between the inside of the UDDI server and outside world. Also, DAL is responsible for exchanging data among internal components of the UDDI server. The agent based DAL can act like a router to transfer incoming data to a suitable component located inside the UDDI server. It is important to note that DAL should be able to realize all Web services standards as UDDI, WSDL, and SOAP.

The main responsibilities of DAL are:

1. Routing incoming and outgoing messages to a suitable internal component or outside world.

2. Exchanging information between the internal components of the UDDI server.

3. Assisting the service requesters in finding suitable service providers by locating and suggesting the most qualified services.

4. Managing all received information regarding performances of an enterprise and updating the qualification of service providers' profiles.

**Inference Engine (IE).**

The current technology of the UDDI server suffers from realizing the meaning of requests. In fact, we need an intelligent component to realize requests both syntactically and semantically. However, agents are *"context-sensitive"* components. They just recognize a term with respect to its syntax rather than semantic. Therefore, there is a need to a paradigm to make agent capable of recognizing terminologies semantically. This paradigm is as *"semantic Web"* or *"ontology"*.

In our model, the ontology consists of an agent based entity named as *"Ontology Agent (OA)"* and knowledge base component named as *"Ontology server"* or *Context Database*, which is responsible for knowledge representation. The ontology server is a knowledge base database that assists OA to reason about the meaning of the underlined terminologies. The need for an ontology server is based on this fact that, due to diversity and wide categories of terminologies, which are adopted by all query languages, it is hard to recognize the meaning of all terms by just reasoning.

The Ontology Agent, or OA, accepts a request or the terminologies that have been used in the request, *"interprets"* it into a specific format, known as *"context information"*, and then submits this information to the ontology server to *"process"*. The process on the context information is carried out by some *"classifications"* of some *"context types"*.

The success or failure result of IE depends on two factors:

1. The interpretation power of OA in producing suitable contextual information.

2. The designing of classification inside the ontology server in order to recognize the meaning of the context information.

Eventually, after interpretation and processing, OA sends back to DAL the meaning of the terms with an understandable format to search at CU.

Discovering a service can be handled in two steps as following:

- *Syntactic Checking.* In this case, the Discovery Agent Layer (DAL) is responsible for matching terminology with the existing one. There are some algorithms to check the similarities of two words such as *"edit distance"* [46]. Edit distance is a method for weighting the difference between two terms. It is based on minimum number of operation in a term to transfer one string into another one. For instance, the edit distance for two words like "Web Service" and "Web-Service" is equal to 1. It means by one operation two strings are equal. There are other methods as well as lexical analyzing method, which we do not discuss about them in this thesis.

- *Semantic Checking.* If the syntactic checking is failed, then the Inference Engine, or IE, is responsible for determining the equivalent semantic of the terms, or requests. The reasoning power, of OA is useful to maintain an up to date ontology server. On other words, OA by discovering a new meaning or synonym for a term saves it into the ontology server as knowledge.

  It is important to note that updating an ontology server is more than just adding a new terminology. Besides saving the new terminologies in the ontology server to have an up to date database, the relations between terms, which are discoverable by OA, should be saved as well. The reasoning power of OA can be defined as considering terminologies themselves as well as the relations among them.

### Core UDDI (CU).

We have considered two subunits inside CU as: *the UDDI-Database* and the *Rating Agent (RA)*. The UDDI database is the same as the current technology of the UDDI registry with the same functionalities. We describe the second proposed component, or the Rating Agent (RA).

The current mechanism of the UDDI server is unable to help the service requesters finding the *"most suitable"* and *"reliable"* service providers. Software agents can be considered as a technology in automation of assessment and management of customers' demands towards selecting a desired service with the best qualification, rating, history, and reputation.

In our model, RA provides some information in terms of qualification of services for assisting service requesters in choosing the more reliable business partners. RA by expanding the UDDI's knowledge of existing services makes it possible evaluating the quality of underlined Web services. RA updates the rating database by evaluating some parameters or measures such as availability, performance, reliability, and response time. The rating information is accessible for any service requester.

Generally in measuring the reliability of services, the following are considered:

1. *Reputation and Rating*, which is a feedback from service requesters to show the

ratio of their satisfactions of offered services. The rating parameter is a statistical measure to give a clue to customer for decision making purposes. From mathematical point of view, a parameter such as reputation or rating can be considered as a set of attributes known as a *"vector"*. Thus by defining some *"globally known standard"* attributes for the vector, clients by either submitting or receiving, achieve information about services. Customers can consider a *"threshold"* value for each attribute of rating vector. Then, checking the eligibility of services will be as checking the attributes of the rating vector with the desired threshold values.

In our model customers after getting services from service providers submit their opinions about the service providers to the UDDI server by interacting with DAL and consequently with RA. RA by receiving some information about service providers and converting them to a *"measurable"* parameter or vector, updates the service providers' profiles, which are saved on the UDDI server. This information assists service requesters in future discoveries.

2. *Response Time*, which is amount of time to receive or deliver services. Communication and feedback time can be considered as parts of this parameter as well. As we mentioned in our agent oriented Web services model, for each goal or task, which is submitted to a partner's Web services, a corresponding *"Controller Agent"* at the Agent Based Controller (ABC) will be created. Besides checking and monitoring the status of task, the controller agent reports the *"response time"* for each task accomplished by an enterprise as well. Consequently, the agent based controller ABC sends this information to PAL and finally to the agent based UDDI. RA interprets the received information as statistical data

and updates the service providers' profiles.

## 3.4 Summary

In this chapter we proposed a Web services / agent based model for enterprise collaboration. We explained the existing problems with current technology of Web services used in enterprise collaboration. We defined an agent based internal architecture for each individual enterprise as well as the UDDI registry entry. We proposed our agents layers on the top of legacy systems of enterprises as well as the UDDI server, at which all interactions between enterprises or the UDDI server and outside world will be carried out by the defined agent layers. Moreover, we addressed the rationale to have such an agent based model for enterprises and the UDDI server, which are supposed to involve in a virtual enterprise.

# Chapter 4

# Software Prototype System Design and Implementation

This chapter presents the design and implementation issues of the proposed architecture and experiences gained from the experimentation with Web services through agents point of view. It also highlights the levels of implementation representing different components of the system. We describe the architectural design issues at enterprises, Web portal, and client sides. At the end, the chapter presents a simple scenario, implemented by the prototype.

## 4.1 System Design

Web services and software agents together provide an integrated solution towards creating an Internet based collaborative environment. Consequently, an Internet based collaborative environment facilitates resource sharing and interaction over the network. This section focuses on the important design issues of the proposed model.

### 4.1.1 Challenges

Internet based collaboration allows enterprises to share their resources, integrate their processes, and provide a new service towards satisfying the market requirements. The resources, which are considered under the content of this thesis, consist of a wide variety of categories as physical machines, database, applications, and software procedures, and human experts. So, one challenge of designing a collaborative environment is to share resources over the Internet. Also, privacy of each participant concerning internal policies, capabilities, and general resources through distributed inter-operations has an important impact on designing a collaborative environment.

From the implementation point of view, using Web services as the main technology and software agents as facilitators, provide essential equipments to design an Internet based collaborative environment.

In this chapter, we present a *simplified* implementation version and important design aspects of the proposed model.

### 4.1.2 System Architecture

As the basic requirements, an Internet based collaboration design must clarify some essential aspects: enterprises communication mechanism, and resource locating, resource sharing, inter-operation mechanism, etc. A simplified architecture of an agent based Web services environment for enterprises collaboration is shown in Figure 4.1.

As a simple case, each enterprise has a Web service, which acts as an *agent* to manage the enterprise resources and inter-operate with others. Enterprises by registering and advertising in UDDI, announce their capabilities for offering some resources and involve into a collaborative environment. A Web portal, by discovering related or similar services in UDDI, collects them and provides a new kind of service, as facility. In fact the Web portal behaves as a *"Web portal agent"* to assist an individual customer, or *"user agent"*, who is interested in establishing a collaboration. On other words, the Web portal agent assists the user agent to find its business partners and set up a collaborative environment.

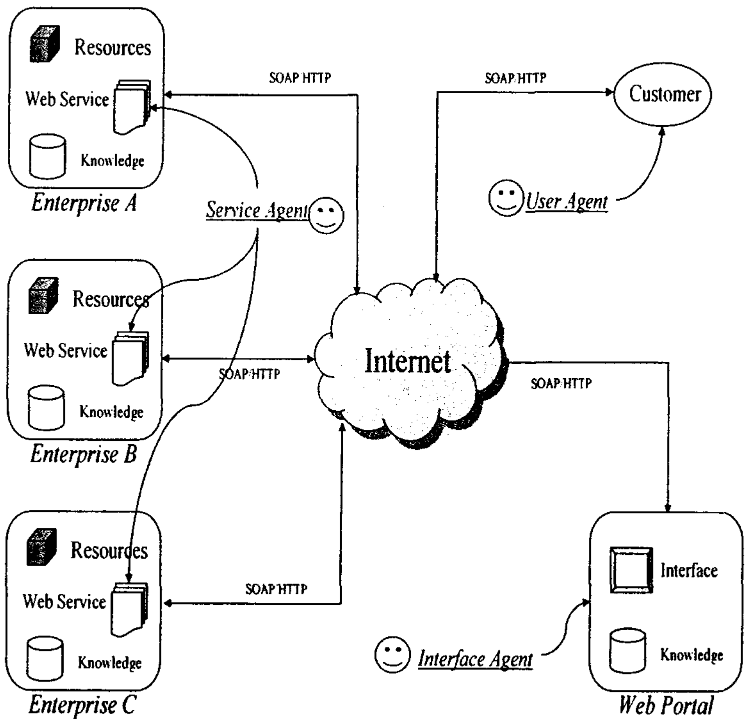


Figure 4.1: A Web services / agent based system architecture for VE

The Web portal, which is considered as a *Web portal agent*, registers in UDDI as well to be discovered by an end user. Consequently, a user agent, interested in finding a business partner discovers the Web portal agent through UDDI. In following we discuss the internal architecture of components involved in a simple collaborative environment, which are depicted in Figure 4.1.

**Customer (User Agent) Architecture.** The internal architecture of each individual customer is depicted in Figure 4.2.

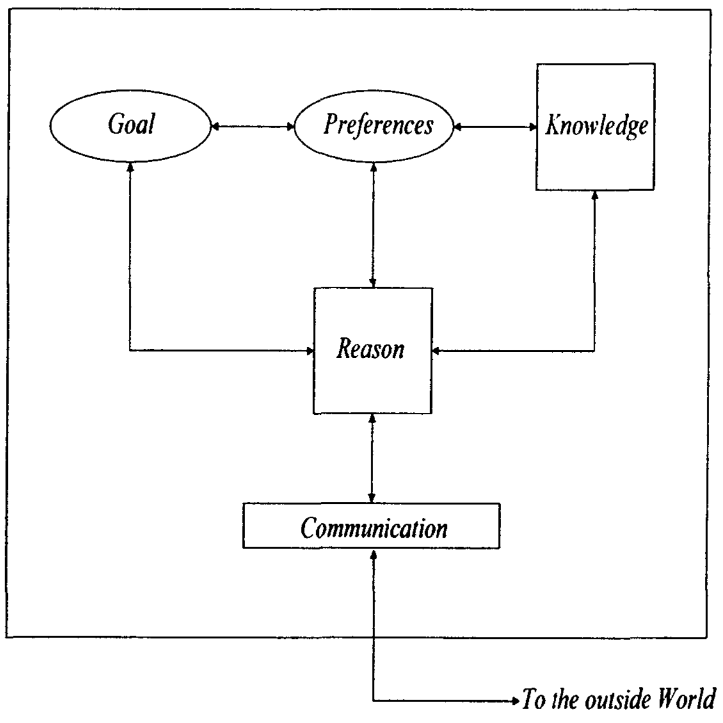


Figure 4.2: The internal system architecture for the customer entity (user agent)

A customer acts as an agent, who needs some specific services and is interested in locating suitable service providers to create a collaborative environment. A user agent is capable of using the Web service communication language, known as SOAP messages over the Internet communication protocol , or HTTP.

A user agent, based on its *knowledge* and *preferences*, defines ultimate goals and acts based on them. A user agent is capable of *reasoning* about defining goals, creating calls for proposals, and accepting offers. In fact, a user agent simply can be a personal assistant application, which by providing some information, assists the end user in finding a suitable business partner.

**Enterprise (Service Agent) Architecture.** An enterprise is an entity, which provides some resources and services and would like to involve in collaboration. Any kind of enterprise has some resources as well as some knowledge. Every enterprise, at design prospective level, looks like Figure 4.3. At communication level, any enterprise has knowledge about Web services communication language, as SOAP, and is able to realize any kind of SOAP messages over the Internet. The main component of any enterprise is its Web service, which acts as an agent and is responsible for realizing incoming SOAP messages and assigning them the existing resources. The mapping mechanism is through the *reasoning* capability of an agent based Web service. A simple Web service consists of following components:

1. ***"Tie"* object**. A server object, which is created for Web service during creation time and is used for communicating with clients' programs. In fact, a tie is a representative object for a Web service.
2. **Interface**. All communication between the Web service and the outside world is handled through interface. An interface acts as a *"wrapper"*, which

encapsulates the complexity of implementation part as well as provides a privacy mechanism to protect the internal structures and policies from the outside view.

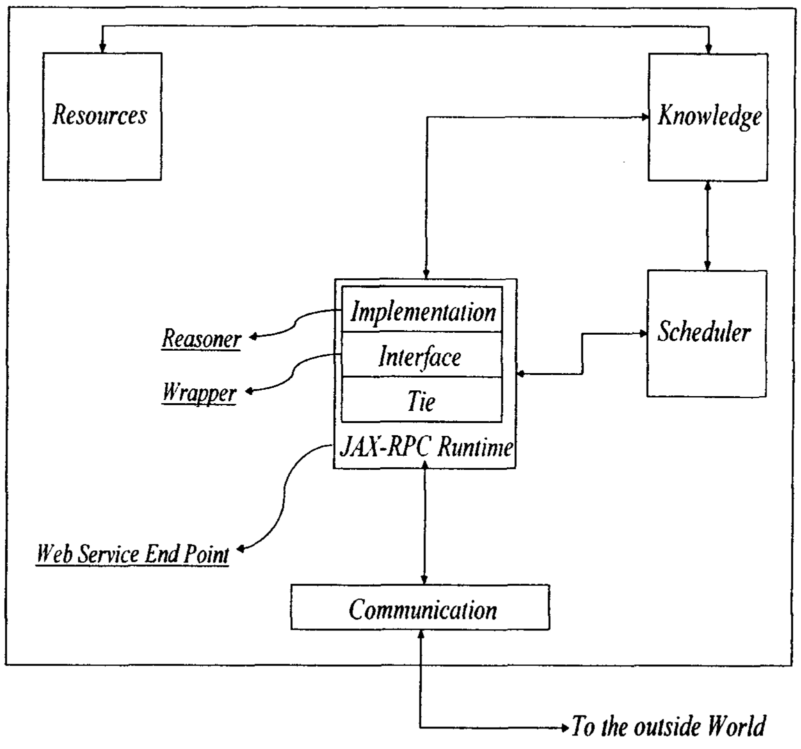


Figure 4.3: The internal system architecture for an individual enterprise entity

3. **Implementation.** The implementation component acts as a *reasoner* part for Web service. The implementation part by achieving incoming requests and based on the knowledge and resource availability, reasons and decides offering a service or refusing the requests. This component consists of some

methods and rules, which are protected by interface component.

In fact, a Web service acts as a service agent responsible for deciding about offering a service that is based on the knowledge and resource availability. In case of offering a service, the process is delegated to a scheduler to generate proposal and to schedule the request as a task. Any Web service to be discovered by requesters may register in UDDI server.

**Web Portal (Agent) Architecture.** A Web portal can be designed either as a Web server or a Web service. In case of a Web *server*, it acts as an agent, which is responsible for locating services or resources for a service requester. However, in case of Web *service*, it acts as an agent as well, which is responsible for either locating resources and services, or offering (composite) services. The internal architecture of a Web portal is depicted in Figure 4.4.

The interface component is a gate, which allows customers to use the Web portal. On other words, the interface is an application, which receives the incoming requests, and delivers them to other internal components. The Web portal has its own knowledge and resources. In case of requesting a service from another enterprise's Web services and creating a composite service, the Web portal needs to acquire an object known as *"stub"* from the partner's Web service, which represents the desired Web service. *"Stub"* is a local object, on the client side, that represents the remote Web service and allows service requesters to communicate with the Web service.

A Web portal should have knowledge about SOAP messages in order to communicate with other Web services. The Web portal acts as a *"coordinator"* as well.

In case of needs to a new service, the Web portal by looking at UDDI, locates the service providers, which provide some desired services, then by analyzing the goal, delegates the sub goals to each service provider and coordinates their processes.

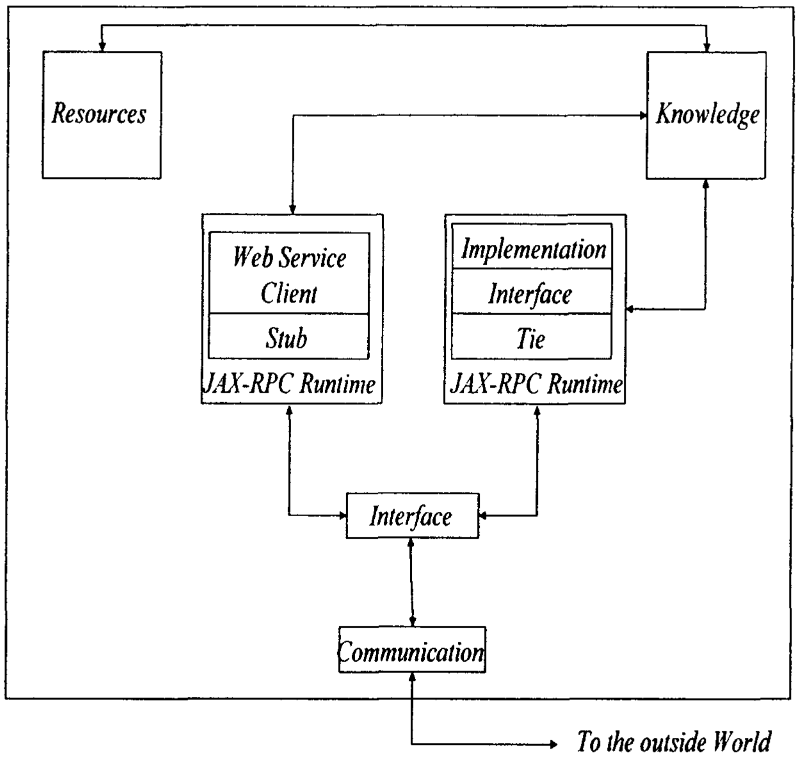


Figure 4.4: The internal system architecture for the Web portal entity (agent)

## 4.2 System Interactions

The implementation among different involved entities describes the behavioral properties of the system. UML sequence diagram has been used to explain the interaction and exchanging messages. Clearly, the messages are based on the structure of SOAP. A general sequence diagram of interaction is shown in Figure 4.5.

In following we describe the interactions among the entities.

**Registering with the UDDI server.** Any service provider, in order to utilize the service discovery by service requesters, may register with the UDDI server. The registration includes any kind of services, either elementary or composite. In the prototype, all enterprises, or service agents, which provide some elementary services, may register in the UDDI server. Moreover, a Web portal may register, in the UDDI server as well. This is important to note that if the Web portal provides a kind of composite service, then it will have its own WSDL files as well.

**End User - User Agent** An end user is responsible for defining goals and their preferences. The user agent assists the end user to choose the most suitable service provider. In fact, the user agent provides the ability for end users to submit their requests to the Web portal agent or to check for any service that would be of an interest. The user agent, by accepting the goal and its performances, contacts either with the Web portal agent or, by looking at UDDI, contacts with service provider directly. Eventually, by receiving the proposals from service providers, the user agent by its knowledge, reasons about the proposals and assist the end user to choose the most appropriate one.

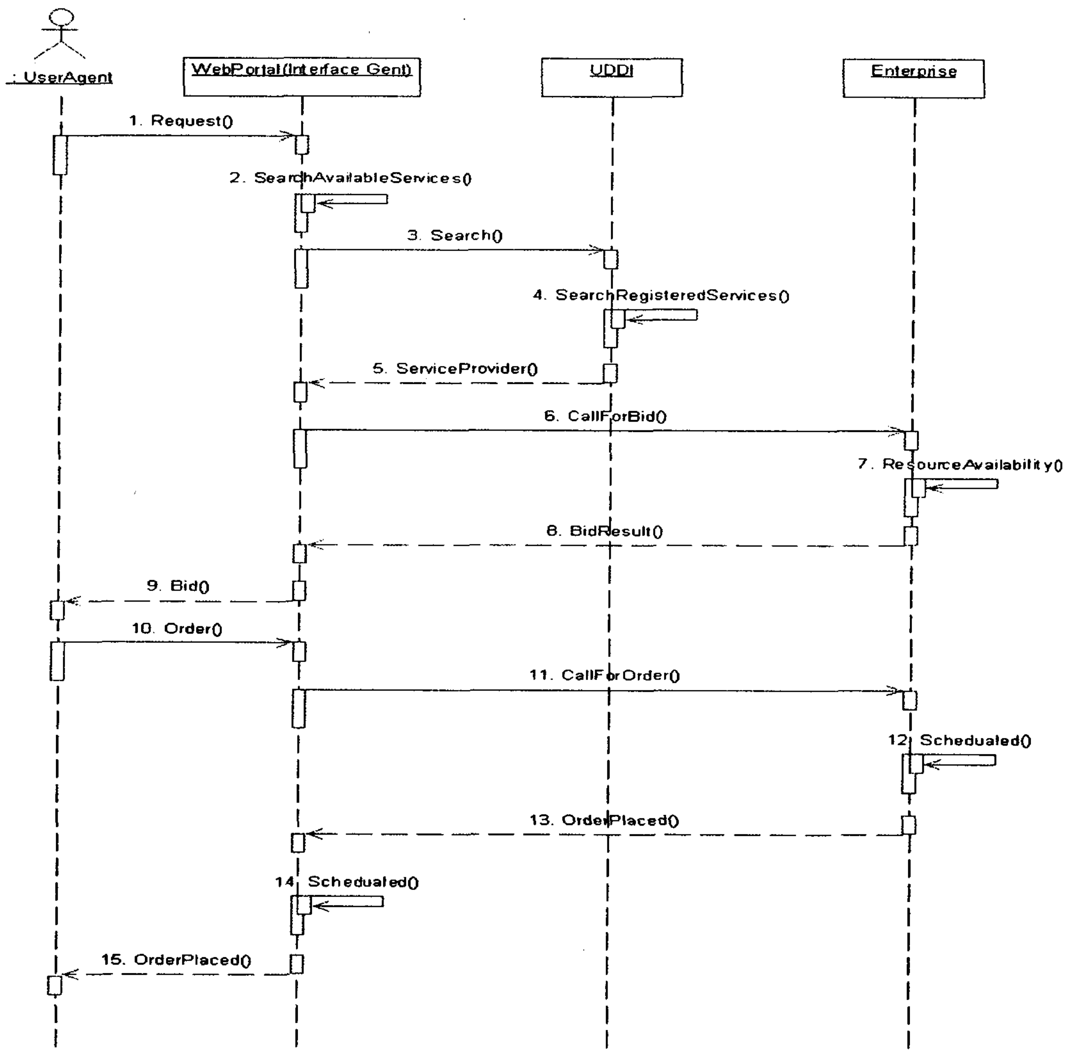


Figure 4.5: The sequence diagram for interaction among entities

**User Agent-Web portal Agent.** A user agent interested in creating a collaborative environment, and sharing resources over the Internet, defines a goal as a requested service and sends it via SOAP messages to the Web portal, or Web portal agent. The goal consists of requested service as well as some preferences. The Web portal, which provides some services, both elementary and composite, receives the SOAP messages and analyzes the request. In case of ability of providing the requested service by the existing one, the Web portal sends back a SOAP message to the user agent including the proposal. The proposal is based on the reasoning and resource availability in the Web portal. Consequently, the user agent reasons about the proposal and decides to order or not. In case of needs to another service provider, the Web portal agent contacts with the UDDI server.

**Web portal Agent-UDDI.** The Web portal, in case of needs to create a composite service, sends a request to UDDI to search and find suitable services, the UDDI based on database knowledge sends back a SOAP message to the Web portal including the service providers and the URL's link to their WSDL files.

**Web portal Agent-Service Agent.** The Web portal by achieving the list of potential services and their providers, or enterprises, sends some SOAP messages to them asking for their proposals. By receiving the SOAP messages, the enterprises based on the availability of resources and knowledge, reason about offering proposals to the Web portal and consequently the user agent. The intelligent Web portal agent, after receiving the proposals and by using some evaluation parameters, such as user preferences, may select the most appropriate service provider as well. Simply the Web portal agent may:

1. Remove all proposals with services that do not match the requested service.
2. Remove all proposals with services that do not satisfy the user preferences.
3. Negotiate with the candidate service provider on behalf of the user agent.

**Service Agent-Web Services.** The service agent assists an individual enterprise to make a most suitable decision for offering a reasonable proposal. The service agent is responsible for reasoning about resources availability, negotiating with service requester, either user agent or Web portal agent, and placing order into the Web service.

## 4.3 A Scenario

The system architecture of the developed prototype is depicted in Figure 4.6.

As it is shown in Figure 4.6 in our prototype there exits some companies, who own some expensive machines that they like offering to customers as available services or resources through the Internet. Each machine has its own *"speed"* and *"price"*. The speed represented by the volume of a machine task, which can be accomplished by the machine in every minute and the price is the cost of work for each machine in every minute. One of the main benefits of implementing a Web service is encapsulating the complexity of services as well as keeping privacy of internal methods, data, and resources. In the developed prototype, customers have no idea about the internal structures, available machines, or internal resources of a specific enterprise. Customers, via the Web portal sends their requests. The Web portal which has knowledge about the location of the URL link of WSDL file for each Web service, delivers the requests to the enterprises' Web services.

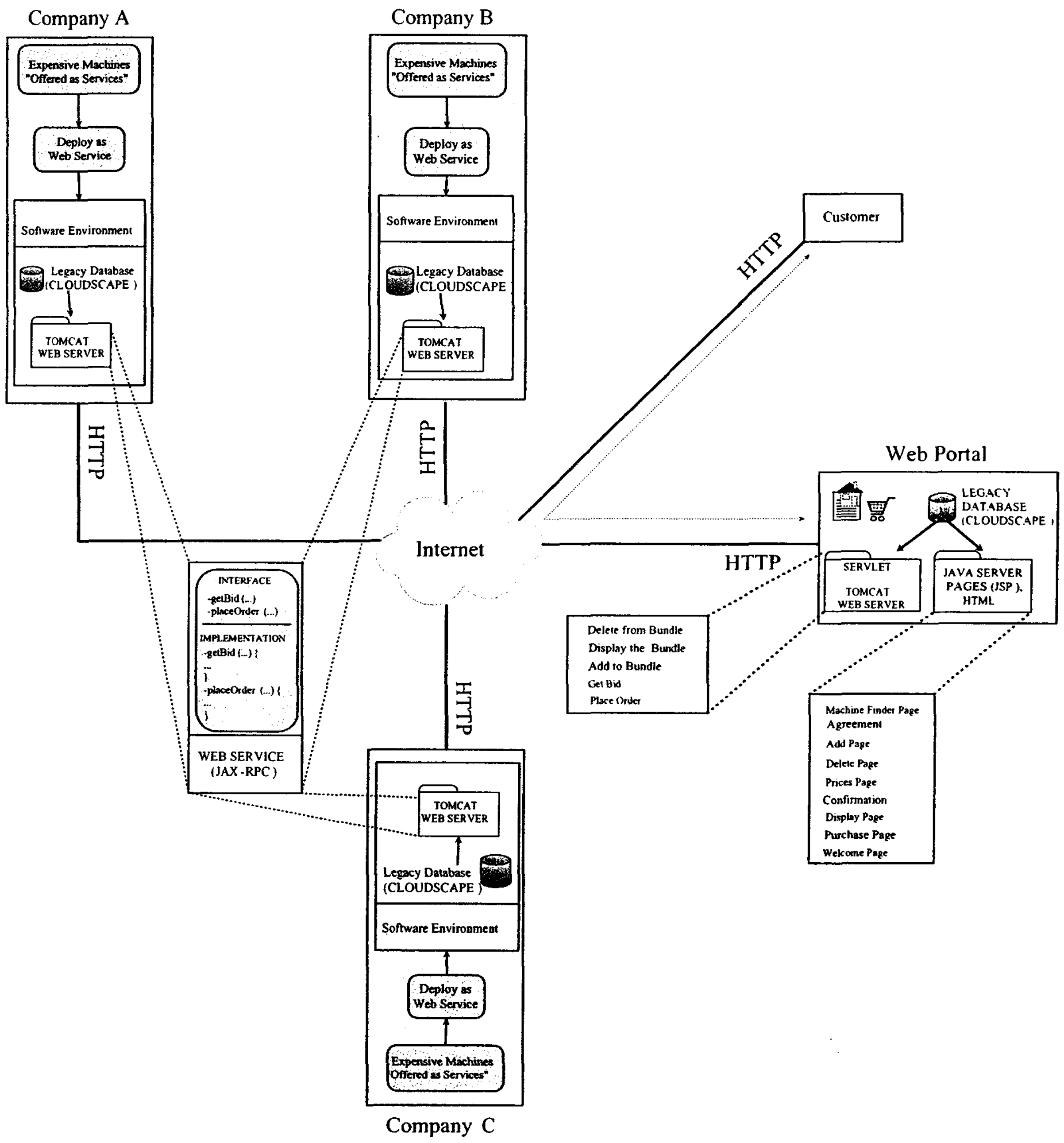


Figure 4.6: The system architecture

Each company has its own legacy database system, which is updated dynamically, and we supposed that the Web portal knows the URL link of the Web services' WSDL files. So, we do not consider the registry and searching level into the UDDI server. The Web portal consists of some methods to define, submit, and get the results back from each Web service. Any customer, by contacting with the Web portal through the Internet, defines any number of jobs, which are packed under one bundle, and then submits the bundle to the Web portal. Consequently, the Web portal by sending the bundle to each of the enterprises, via their Web services, asks them to offering their proposals. Each bundle consists of some tasks, which are defined in terms of machine's type, volumes, and due dates, by which all jobs in the bundle should be completed.

Moreover, we suppose each machine in a company has a special rank that ranks machines with respect to their speeds and cost. Assigning jobs to each machine is based on the rank that results in offering bids to customers with the lowest cost as much as possible.

On the companies sides, each enterprise based on the availability of machines and with respect to the due date of the requested bundle, decides to offer any bid. If the requested bundle can be carried out by the company before the due date, then the enterprise offers a bid. This work is supposed to be done by an agent-based dynamic scheduling system [30], which is beyond the scope of this thesis.

By receiving proposals, the customer chooses the best and most suitable offer and places its order through the Web portal to make contract and service agreement. Making contract is just simply by asking customer's information to keep into the company's database and issuing the job reference numbers to show the order and service agreement.

In our scenario, each Web service of an individual enterprise has two methods as services invoked by customers. One method has been used for generating bids and the other one for placing orders. All Web services have been deployed on company's Web server using Apache Tomcat. The Web portal, is an application to connect to enterprises' Web services to receive data and show the results to the end users. Each customer by browsing the entry page of Web portal requests bids and places an order. Communication between entities are carried out by HTTP.

Detailed Use Case, Class, and Sequence diagrams can be found in [32].

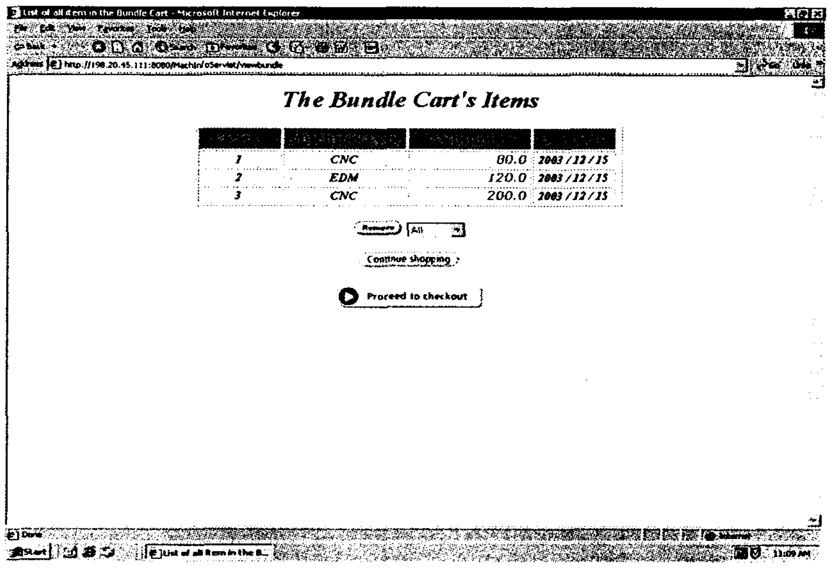
Figure 4.7 depicts the defined tasks in a bundle to send to the enterprises' Web services to call for their bids and Figure 4.8 shows the results of companies' bids.

## 4.4 Prototype Implementation

In order to prove the feasibility of the proposed model, a simple prototype has been developed. The prototype takes advantages of Java language and the Java based toolkit for Web services, known as JAX-RPC. The prototype is a simplified distributed system, which represents integration and resource sharing through a cooperative distributed system. It consists of following entities:

1. A number of enterprises as service providers, capable of providing some services or resources.

2. An agent based Web portal behaving as a gateway, through which end users send their requests to the Web services and the Web portal assists and uses to find a suitable service provider with respect to their preferences.



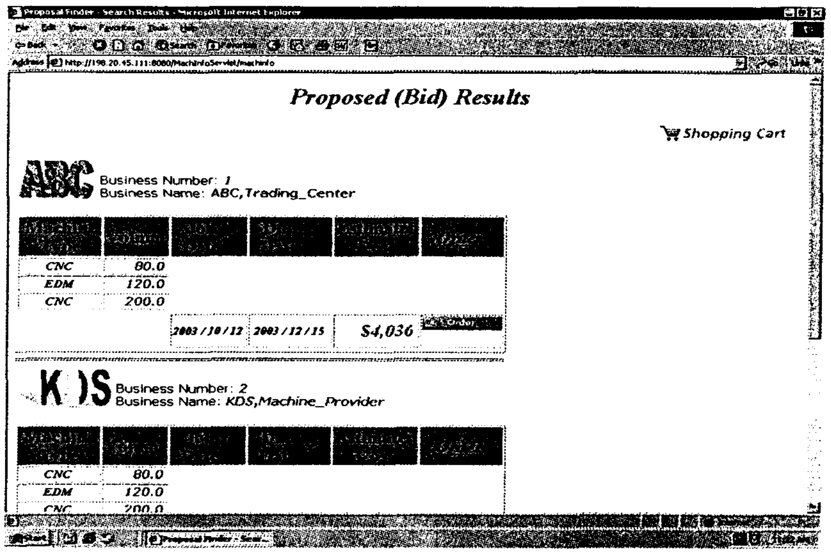
List of all item in the Bundle Cart - Microsoft Internet Explorer

http://198.20.45.111:8080/MachInfoServlet/viewbundle

**The Bundle Cart's Items**

| | | | |
|---|---|---|---|
| 1 | CNC | 80.0 | 2003/12/15 |
| 2 | EDM | 120.0 | 2003/12/15 |
| 3 | CNC | 200.0 | 2003/12/15 |

Figure 4.7: The defined tasks in a bundle (call for bids)



Proposal Finder - Search Results - Microsoft Internet Explorer

http://198.20.45.111:8080/MachInfoServlet/machinfo

**Proposed (Bid) Results**

Business Number: 1
Business Name: ABC,Trading_Center

| | | | | | |
|---|---|---|---|---|---|
| CNC | 80.0 | | | | |
| EDM | 120.0 | | | | |
| CNC | 200.0 | | | | |
| | | 2003/10/12 | 2003/12/15 | $4,036 | Order |

Business Number: 2
Business Name: KDS,Machine_Provider

| | | | | | |
|---|---|---|---|---|---|
| CNC | 80.0 | | | | |
| EDM | 120.0 | | | | |
| CNC | 200.0 | | | | |

Figure 4.8: The result of enterprises' bids

### 4.4.1 Tools Used for Prototype Implementation

The prototype has been developed on some popular Web programming tools and languages under Windows NT/2000 environment. The most demanded language in today's technology known as JAVA has been used for the whole project. The following tools have been used in developing the prototype:

- Java API for XML-based RPC known as **JAX-RPC**, to create and deploy all Web services. This product is available through *Java Web Service Developer Pack* (current version 1.2) under the Sun's Web site *www.java.sun.com.*
- Java Developer Kit (JDK 1.4), to develop almost the whole prototype.
- Cloudscape database, to create the databases both on the Web Services and the Web portal sides. This tool is available through the Sun's Web site as well.
- Java Database Connectivity (JDBC), to save and retrieve data.
- Java Servlet Technology, to create some server side programs.
- Java Server Pages (JSP) and HTML to create some end user Web based interfaces.
- Apache Tomcat, to deploy the Web services and Servlets.

#### The JAX-RPC

JAX-RPC is a collection of procedures that can be called over the Internet. The remote procedural call, or RPC, nature of JAX-RPC, makes it an easy tool to use and deploy Web services and consequently, very usable for developers. A client of a Web service simply makes Java method calls, and all internal operations are carried out automatically. On the server side, The Web service simply implements the services and by using interface to encapsulate the complexity of implementation to calling clients. The JAX-RPC enables developers to create SOAP based inter-operable and portable Web services. JAX-RPC enables clients to invoke the Web services developed across heterogeneous platforms. Complexity of implementation of the Web services is encapsulated by some runtime mechanisms and interfaces. Therefore, deploying a Web service is an easy task.

The architecture of JAX-RPC is depicted in Figure 4.9. As it is clear from the Figure JAX-RPC requires SOAP over HTTP for inter-operability (Web service communication language).

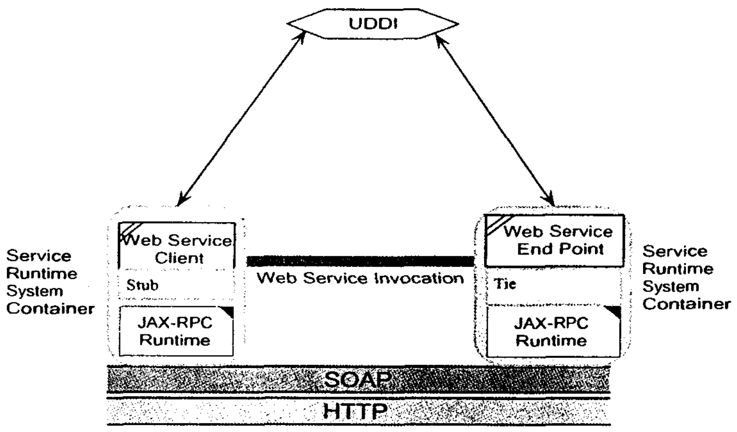


Figure 4.9: The JAX-RPC architecture

In JAX-RPC, a remote procedure call is handled by XML-based protocol such as SOAP. All SOAP messages, which consist of header and body are transmitted over the HTTP protocol.

### 4.4.2 Software Components

As we have already mentioned in the previous section, In our scenario there are some companies who has some expensive machines offered as services to customers. A client who is interested in using these machines as services by invoking and asking for their proposals decides in choosing the suitable enterprise as a business parter. The configuration of the prototype is shown in Figure 4.10:

Each service provider has its own: database, services, configuration files, etc. The underlined Web services are developed by JAX-RPC deployed on the Apache Tomcat Web container. The database for both the Web services and the Web portal application is created by *Cloudscape database.*

The Web portal is created by Java Servlets, deployed on Apache Tomcat Web container as well. The server side programs or Servlets are invoked by Java Server Page (JSP) and HTML programs. The Web portal has its own database, which is created by *Cloudscape* database.

Figure 4.11 shows a sample Web service, which has been developed for some virtual companies.

Consequently, the following source code shows the WSDL (Web service description language) file for one of the Web services. The WSDL file has been created by the Web service to describe the services, methods, and passing parameters, return values, etc. and generally interacting methods with the Web service.

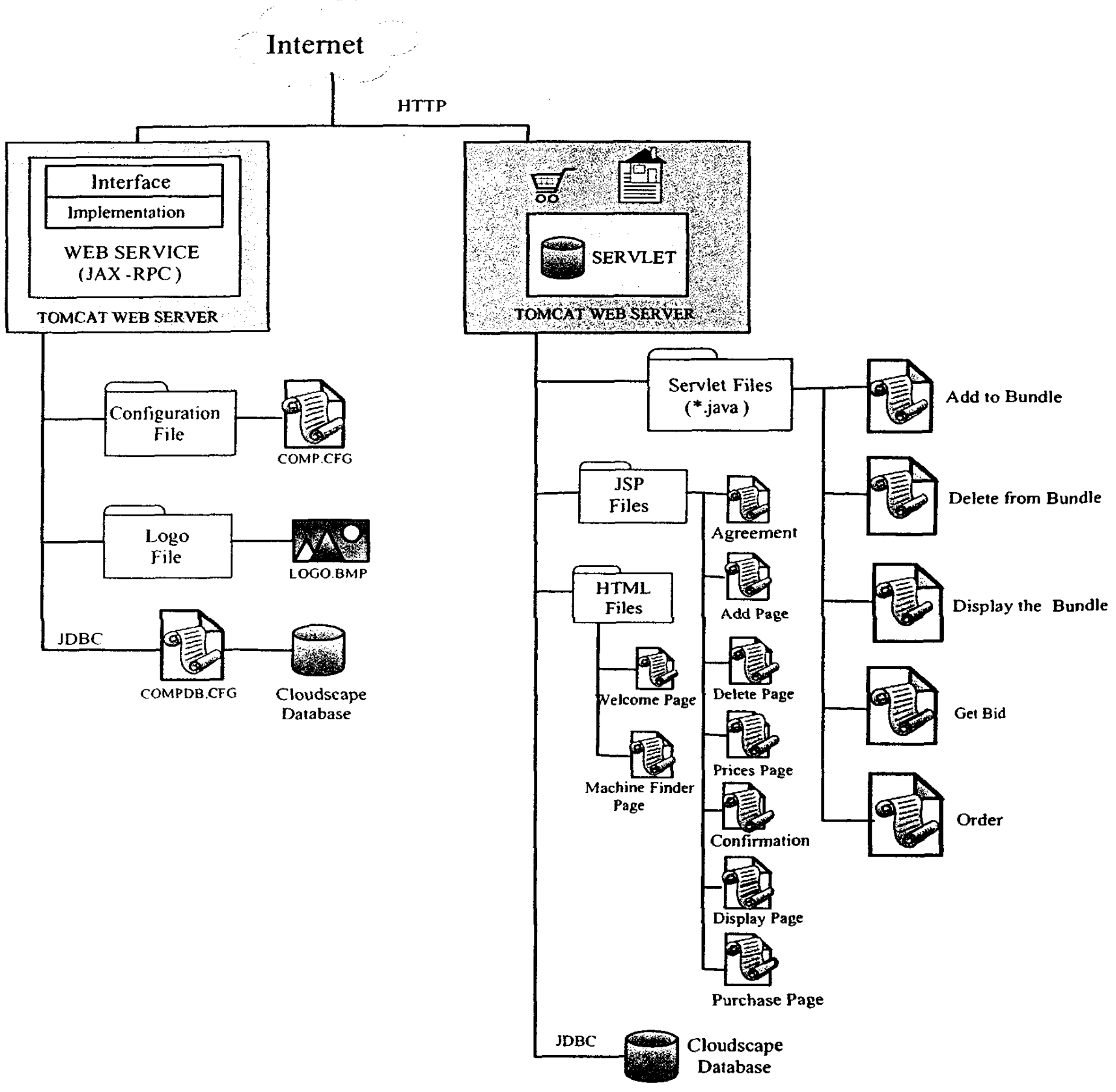
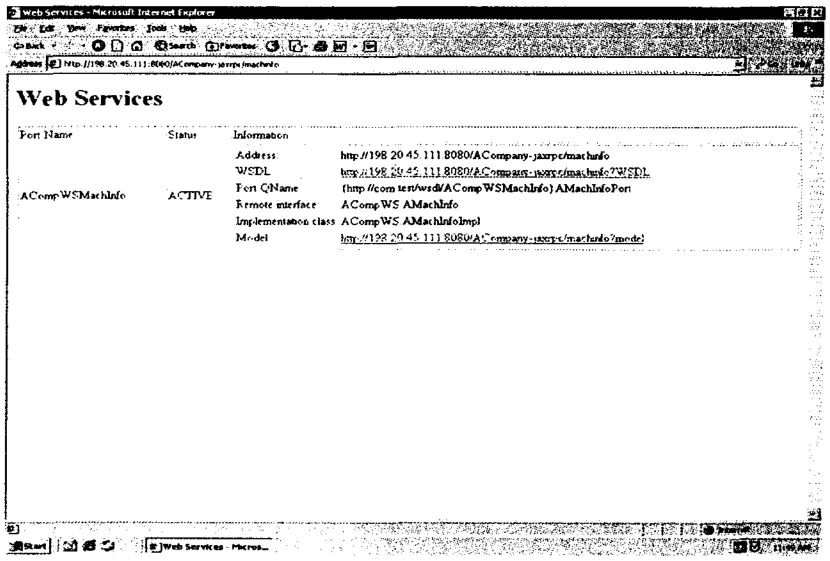


Figure 4.10: The implementation architecture of the prototype



Web Services

| Port Name | Status | Information | |
|---|---|---|---|
| ACompWSMachInfo | ACTIVE | Address: | http://198.20.45.111:8080/ACompany-jaxrpc/machinfo |
| | | WSDL | http://198.20.45.111:8080/ACompany-jaxrpc/machinfo?WSDL |
| | | Port QName | {http://com.test/wsdl/ACompWSMachInfo}AMachInfoPort |
| | | Remote interface | ACompWS.AMachInfo |
| | | Implementation class | ACompWS.AMachInfoImpl |
| | | Model | http://198.20.45.111:8080/ACompany-jaxrpc/machinfo?model |

Figure 4.11: The Web service for an enterprise

```
<? xml version="1.0" encoding=" UTF-8" ?>
- <definitions xmlns="http:// schemas.xmlsoap.org /wsdl/"
  xmlns:tns="http:// com.test/wsdl/ACompWSMachInfo "
  xmlns:xsd ="http:// www.w3.org/2001/XMLSchema"
  xmlns:soap="http:// schemas.xmlsoap.org /wsdl/soap/"
  xmlns:ns2="http:// com.test/types/ACompWSMachInfo "
  name=" ACompWSMachInfo "
  targetNamespace ="http:// com.test/wsdl/ACompWSMachInfo ">
- <types>
- <schema xmlns="http:// www.w3.org/2001/XMLSchema"
  xmlns:tns="http:// com.test/types/ACompWSMachInfo "
  xmlns:wsdl="http:// schemas.xmlsoap.org /wsdl/"
  xmlns:xsi="http:// www.w3.org/2001/XMLSchema-instance"
  xmlns:soap-enc="http:// schemas.xmlsoap.org /soap/encoding/"
  targetNamespace ="http:// com.test/types/ACompWSMachInfo ">
  <import namespace ="http:// schemas.xmlsoap.org /soap/encoding/"/>
- < complexType name=" JobPriceBid ">
- <sequence>
  <element name=" compAddress " type="string" />
  <element name=" compCity " type="string" />
  <element name=" compCountry " type="string" />
  <element name=" compDes" type="string" />
  <element name=" compID " type=" int" />
  <element name=" compLogo" type="string" />
  <element name=" compState " type="string" />
  <element name=" compTel" type="string" />
  <element name=" compWebsite " type="string" />
  <element name=" dueDate " type="string" />
  <element name=" jobampm " type=" int" />
  <element name=" jobfindate " type="string" />
  <element name=" jobfintime " type="string" />
  <element name=" joblength " type=" int" />
  <element name=" jobprice " type="double" />
  <element name=" machtype " type="string" />
  <element name="volume" type="double" />
</sequence>
</complexType>
- < complexType name=" MachCustomer ">
- <sequence>
  <element name="city" type="string" />
  <element name="country" type="string" />
  <element name=" creditCardNumber " type="string" />
  <element name=" emailAddress " type="string" />
  <element name=" firstName " type="string" />
  <element name=" lastName " type="string" />
  <element name="state" type="string" />
  <element name=" streetAddress " type="string" />
  <element name=" zipCode" type="string" />
</sequence>
</complexType >
- < complexType name=" JobRefOrder ">
- <sequence>
  <element name=" jobampm " type=" int" />
  <element name=" jobcost" type="double" />
  <element name=" jobduedate " type="string" />
  <element name=" jobfindate " type="string" />
  <element name=" jobfintime " type="string" />
  <element name=" joblength " type=" int" />
  <element name=" jobmachtype " type="string" />
  <element name=" jobnumber " type=" int" />
  <element name=" jobvolume " type="double" />
</sequence>
</complexType >
</schema>
</types>
- <message name=" AMachInfo_getBid ">
  <part name="String_1" type=" xsd :string" />
  <part name="double_2" type=" xsd :double" />
  <part name="String_3" type=" xsd :string" />
</message>
- <message name=" AMachInfo_getBidResponse ">
  <part name="result" type=" ns2:JobPriceBid " />
</message>
- <message name=" AMachInfo_placeOrder ">
  <part name="String_1" type=" xsd :string" />
  <part name="double_2" type=" xsd :double" />
  <part name="String_3" type=" xsd :string" />
  <part name=" MachCustomer_4" type=" ns2:MachCustomer " />
</message>
- <message name=" AMachInfo_placeOrderResponse ">
  <partname="result" type=" ns2:JobRefOrder " />
</message>
- < portType name=" AMachInfo ">
- <operation name=" getBid " parameterOrder ="String_1 double_2 String_3">
  <input message=" tns:AMachInfo_getBid " />
  <output message=" tns:AMachInfo_getBidResponse " />
</operation>
- <operation name=" placeOrder " parameterOrder ="String_1 double_2 String_3 MachCustomer_4">
  <input message=" tns:AMachInfo_placeOrder " />
  <output message=" tns:AMachInfo_placeOrderResponse " />
```

```
</operation>
</portType >
- <binding name=" AMachInfoBinding " type=" tns:AMachInfo ">
- <operation name=" getBid ">
- <input>
  <soap:body encodingStyle ="http:// schemas.xmlsoap.org /soap/encoding/" use="encoded"
namespace="http://com.test/wsdl/ACompWSMachInfo " />
</input>
- <output>
  <soap:body encodingStyle ="http:// schemas.xmlsoap.org /soap/encoding/" use="encoded"
namespace ="http://com.test/wsdl/ACompWSMachInfo " />
</output>
  <soap:operation soapAction ="" />
</operation>
- <operation name=" placeOrder ">
- <input>
  <soap:body encodingStyle ="http:// schemas.xmlsoap.org /soap/encoding/" use="encoded"
namespace ="http://com.test/wsdl/ACompWSMachInfo " />
</input>
- <output>
  <soap:body encodingStyle ="http:// schemas.xmlsoap.org /soap/encoding/" use="encoded"
namespace ="http://com.test/wsdl/ACompWSMachInfo " />
</output>
  <soap:operation soapAction ="" />
</operation>
  <soap:binding transport="http:// schemas.xmlsoap.org /soap/http" style=" rpc " />
</binding>
- <service name=" ACompWSMachInfo ">
- <port name=" AMachInfoPort " binding=" tns:AMachInfoBinding ">
  <soap:address xmlns:wsdl="http:// schemas.xmlsoap.org /wsdl/"
location="http://198.20.45.111:8080/ ACompany -jaxrpc/machinfo " />
</port>
</service>
</definitions>
```

## 4.5 Summary

This chapter described the design and implementation issues of an agent based Web services model in a collaborative environment. The main entities involved in creating a collaborative environment should be able to realize the SOAP messages in order to interact. Each Web service behaves as an agent, which is able to reason about circumstances based on some knowledge and the availability of resources. We presented the system architecture of the prototype. We developed some Web services (service agents), representing enterprises and a Web portal (agent) to exchange data with the Web services.

# Chapter 5

# Conclusion and Future Work

This chapter concludes the thesis and discusses our future work.

## 5.1 Conclusion

Enterprises in order to carry out customers' requests need to collaborate with each others and share their resources, data, technologies, and even human skills. Enterprise collaboration, permanent or temporary, requests for a higher level of technology, which allows enterprises to integrate their applications regardless of platforms, data structures, or models.

Web services, known as the new generation of distributed computing, are self describing software components that offer some facilities to integrate applications and carry out workflows. Web services are discoverable for service requesters by registering through the UDDI entries as available services. The UDDI registry is a searchable database, which provides some information regarding service providers as well as a link to the Web Service Description Languages (WSDL) for each Web service. Thus, by analyzing the content of WSDL's, service requesters find out how to use and interact with Web services.

However, the current technology of Web services is still in its infancy stages. It suffers from not only in the UDDI registry, but also adopting in a collaborative environment. On the UDDI side, the search strategies are *"syntactically"*. Moreover, there is no mechanism in assisting service requesters to locate suitable and reliable services. On the Web service side, the service cannot analyze all incoming requests. In fact, Web services are some *"passive"* components that do not have any knowledge about the environment and especially virtual collaboration.

In this thesis, we discuss the feasibility of using software agents to overcome some shortcomings of Web services technology for adopting into a collaborative environment. Agents are software components capable of reasoning, which achieve their knowledge from the environment. The thesis proposes a Web services / agent-based approach towards setting up a virtual environment. Also an agent based model for the UDDI registry has been proposed.

On the UDDI side, the thesis defines some components such as *Discovery Agent Layer (DAL)*, *Inference Engine (IE)*, and *Core UDDI (CU)*. The discovery agent layer is an interface between the UDDI registry and outside world. The Inference Engine is responsible for realizing user's requests semantically. The Core UDDI consists of some other components including the current technology of the UDDI.

On the enterprise side, some other components have been defined such as *Central Management Unit (CMU)* to define a dynamic workflow, *Proxy Agents Layer (PAL)* as an interface between the service and outside world, *Inference Engine (IE)* responsible for analyzing the incoming requests semantically, and *Core Services (CS)* including the current technology of Web services.

We argued the needs for software agents to assist the Web services offering more intelligent services in a collaborative environment. In summary, the major contributions of this thesis include:

- A Web services / Agent-Based model for virtual enterprise. The designed units are:
    1. An agent-based unit for defining a dynamic goal-based model for workflows.
    2. An agent-based unit for coordinating the designed workflow among participants in the collaboration environment.
    3. An agent-based unit for controlling and monitoring the status of delegated tasks to business partners.
    4. An agent-based unit for ontological purposes.
    5. A Web service-based unit for resources sharing purposes.
- An agent-based model for UDDI registry. The designed units are:
    1. An agent based for rating and ranking of registered Web services.
    2. An agent-based unit for ontological purposes.

## 5.2 Future Work

In this thesis we proposed a Web services / agent based approach applicable in a virtual enterprise. However, Web services paradigm is still in its infancy stages and needs lots of R & D efforts. The following list *must* be answered in future work:

- *Dealing With False Rating Issues.* Fake users or services may submit falsely improving the reputation of a particular service. Consequently, malicious users or services may submit bad ratings for services and as a result falsely decreasing a service reputation. This challenge seems being solvable by considering some security issues such as *"threshold schemas"*. As a primitive stage, we propose using some threshold schemas used in *"metering schemas"*. By achieving a number of *"tokens"*, any legal service requester, besides submitting some information regarding rating, presents a token to the UDDI server. In fact, metering schemas are mathematical mechanisms for visiting a server, in which a server by getting visited by a specific numbers of clients, or threshold, can be paid. Therefore, by adopting metering schemas to the UDDI server, the problem of false rating can be solved.

- *Creating a Standard Ontology Server.* Apparently, in order to deliver successful services to customers, the following question needs to be answered first:

  *"How to realize the users' requirements?"*

  In order to realize and offer services to end users, a Web service or the UDDI server should have a *"unified"*, *"standard"*, and *"complete"* ontology server, by which ontology agent interprets and reasons about the user requirements.

- *Creating a Unified and Standard Attributes for Rating Vector.* The challenging question is:

  *"How can we measure the rating of a service?"*

  In case of numerical attributes and quantities values, defining and unifying vector is not a big issue. However, while dealing with quality attributes, it seems being a challenging work, which needs more investigation. Thus, we need to provide a standard model to define and unify rating vectors at different domains.

- *Creating a Genetic Algorithm for Rating.* The rating algorithm should be applicable for different domains and give a standard and unified solution for different UDDI servers. Moreover, the algorithm should be dynamic enough to accept *"user defined"* attributes for rating and take them into account.

- *Dealing With Security Issues.* Security is always a withdraw from implementing a model. Without secure transactions, enterprises never involve in any kind of business, especially E-business. In our model there are still some concerns in security such as security of Web services itself.

  The other issue is privacy, by which companies should be able to protect their internal workflows and data. A proxy agent should be intelligent enough to prevent malicious requests to get inside the enterprise. Due to firewall friendly nature of Web services, current technology of firewall is not suitable for security issues of Web services. Probably an important issue in Web services technology is *how to secure Web services?*.

- *Reliability of Goal-Based Work Flow Designer Unit.* The question is:

  *"How to design a goal-based workflow designer?"*

  In a static workflow, due to its rule-based configuration assigning tasks to participants and replacing them is a straightforward process. In case of dynamic and "goal-based"situation, which have been proposed in this thesis, assigning and replacing a partner is a very difficult. Changing a partner should be fast and straightforward to handle the whole process in time.

# Bibliography

[1] M. Aoyama, S. Weerawarana, H. Maruyama, C. Szyperski, K. Sullivan, and D. Lea, *Web Services Engineering: Promises and Challenges*, Proceeding of the 24th international conference on Software engineering, May 2002, pp. 647–648.

[2] B. Benatallah, M. Dumas, M. C. Fauvet, F. A. Rabhi, and Q. Z. Sheng, *Overview of Some Patterns for Architecting and Managing Composite Web Services*, ACM SIGecom Exchanges **3** (2002), no. 3, 9–16.

[3] W. Brenners, R. Zarnekow, and H. Wittig, *Intelligent Software Agents*, Springer Verlag, Berlin, Heidelberg, 1998.

[4] P. A. Buhler, J. M. Vidal, and H. Verhagen, *Adaptive Workflow = Web Services + Agents*, Proceeding of the International Conference on Web Services, Las Vegas, Nevada, USA (2003), 131 - 137.

[5] L. M. Camarinha-Matos and H. Afsarmanesh, *Infrastructures for Virtual Enterprises*, Kluwer Academic Publisher, Norwell, Massachusetts, 1999.

[6] A. Ceponkus, *Business Transcation Protocol*, 1 ed., OASIS Committee Specification, June 2002.

[7] J Chang and C. Scott, *Agent-based workflow:TRP support Environment(TSE)*, Computer Networks and ISDN Systems (1996), 1501–1511.

[8] Z. Cheng, M. P. Singh, and M. A. Vouk, *Composition Constraints for Semantic Web Services*, WWW2002 Workshop on Real World RDF and Semantic Web Application (2003).

[9] J. Dale, S. Willmott, and B Burg, *Agentcities: Challenges and Deployment of Next-Generation Service Environment*, In proceeding of the Pacific Rim Intelligent Multi-Agent Systems conference, Tokyo, Japan, August 2002.

[10] H. M. Deitel, P. J. Deitel, B. DuWaldt, and L. K. Trees, *Web Services a Technical Introduction*, first ed., Deitel Developer Series, Prentice Hall, New Jersey, 2003.

[11] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman, *On Cooperation in Multi-Agent Systems*, The Knowledge Engineering Review **12** (1997), no. 3, 309–314.

[12] V. Ermolaev, *Agent-Mediated Web Service Compostition for Rational Information Retrieval*, European Communities, 2002- 2004, Multiple Project Consortium (2003).

[13] M. Hardwick and R. Bolton, *The Industrial Virtual Enterprises*, Communication of the ACM **40** (1997), no. 9, 59–60.

[14] N. R. Jennings, *Coordination Techniques for Distributed Artificial Intelligence*, In Foundation of Distributed Artificial Intelligence (1996), 187–210.

[15] N. R. Jennings, P. Faratin, M. J. Johnson, T. J. Norman, P. O'Brien, and M. E. Wiegand, *Agent-based business process management*, International Journal of Cooperative Information Systems. (1996), 105–130.

[16] N. R. Jennings, T. J. Norman, and P. Faratin, *ADEPT: An Agent-Based Approach to Business Process Management*, ACM SIGMOD Record (1998).

[17] T. Jin and S. Goschnick, *Utilizing Web Services in an Agent Based Transaction Model (ABT)*, Workshop on Web Services and Agent-based Engineering (2003), Melbourn, Australia.

[18] H. Kuno and A. Sahai, *My Agent wants to talk to your service: Personalizing Web Services through Agents*, Challenges in Open Agent Systems (2002), Bologna, Italy.

[19] M. Laukkanen and H. Helin, *Composing Workflows of Semantic Web Services*, Workshop on Web Services and Agent-based Engineering (2003), Melbourn, Australia.

[20] M. Lyell, L. Rosen, M. Casagni-Simkins, and D. Norris, *On Software Agents and Web Services:Usage and Design Concepts and Issues*, AAMAS'2003 Workshop on Web Services and Agent-based Engineering (2003), Melbourn, Australia.

[21] Z. Maamar, Q. Z. Sheng, and B. Benatallah, *Interleaving Web Services Composition and Execution Using Software Agents and Delegation*, AAMAS'03 Workshop on Web Services and Agent-based Engineering (2003).

[22] E. M. Maximilein and M. P. Singh, *Reputation and Endorsement for Web Services*, ACM SIGecom Exchanges **3** (2001), no. 1, 24 – 31.

[23] M. Maximilein and M. P. Singh, *Conceptual Model of Web Service Reputation*, ACM SIGMOD Record **31** (2002), no. 4, 36 – 41.

[24] B. Medjahed, B. Bentallah, A. Bouguettaya, A. H. H. Ngu, and A. K. Elmagarmid, *Business-to-Business interactions: issues and enabling technologies*, The VLDB Journal - The international Journal on Very Large Data Bases **12** (2003), no. 1, 59–85.

[25] A. Mowshowitz, *Virtual Organization*, Communication of the ACM **40** (1997), no. 9, 30–37.

[26] T. Murata, *Petri Nets: Properties, Analysis, and Applications.*, Proceeding of the IEEE **77** (1989), no. 4, 541–580.

[27] S. Narayanan and S. A. Mcllraith, *Simulation, Verification and Automated Composition of Web Services*, Proceeding of the eleventh international conference on World Wide Web (2002), 77 - 88.

[28] K. Palacz and D. C. Marinescu, *An agent-based workflow management system.*, Technical report (1999), Computer Science Department, Purdue University.

[29] D. Richards, S. Van Splunter, F. M. T. Brazier, and M. Sabou, *Composing Web Services using an Agent Factory*, AAMAS'2003 Workshop on Web Services and Agent-based Engineering (2003), Melbourn, Australia.

[30] W. Shen, *Distributed Manufacturing Scheduling Using Intelligent Agents*, IEEE Intellegnt Systems **17** (2002), no. 1, 88–94.

[31] W. Shen, D. H. Norrie, and J. P. A. Barthes, *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*, Taylor and Francis, New York, USA, 2001.

[32] A. Siami Namin and W. Shen, *Application of Web Services and Software Agents to Inter-Enterprise Collaboration*, (2003), Controlled Technical Report, NRC-IMTI.

[33] M. P. Singh, *Distributed Enactment of Multiagent Workflows: Temporal Logic for Web Service Composition*, Proceeding of the second international joint conference on Autonomous agents and multiagent systems (2003), 907 - 914.

[34] S. V. Splunter, M. Sabou, F. M. T. Brazier, and D. Richards, *Configuring Web Services, using Structuring and Techniques from Agent Configuration*, Proceeding of the 2003 IEEE/WIC internationl Conference on Web intelligence (2003), Halifax, Canada.

[35] W. J. Van Den Heuvel and Z. Maamar, *Moving Toward A Framwork to Compose Intelligent Web Services*, Communication of the ACM **46** (2003), no. 10, 103 – 109.

[36] Website, http://www.niiip.org/.

[37] Website, http://www.agentcities.org/.

[38] Website, http://www.agentcities.org/.

[39] Website, http://www.zsu.zp.ua/racing/.

[40] Website, http://www.agentcities.net/.

[41] Website, http://www.daml.org/language/.

[42] Website, http://www.fipa.org/.

[43] Website, http://wiki.cs.uiuc.edu/cs427/Fipa-OS.

[44] Website, http://www.fipa.org/specs/fipa00001/.

[45] Website, http://www.fipa.org/specs/fipa00012/.

[46] A. Williams, A. Padamanabhan, and M. B. Blake, *Local Consensus Ontologies for B2B-Oriented Service Composition*, Proceeding of the second international joint conference on Autonomous agents and intelligent systems (2003), 647 – 654.

[47] S. Willmott, J. Dale, J. Picault, M. Somacher, S. Poslad, J. J. Tan, I. Constantinescu, and D. Bonnefoy, *The Agentcities Network Architecture*, In proceeding of the first International workshop on challenges in Open Agent Systems, July 2002.

[48] M. J. Wooldridge and N. R. Jennings, *Intelligent Agents, Lecture Nots in Artificial Intelligence*, Springer Verlag, 1995.

[49] S. T. Yuan and K. J. Lin, *WISE - Building Simple Intelligence into Web Services*, ACM SIGMOD Record **31** (2002), no. 4.

[50] L. Zeng, A. Ngu, B. Benatallah, and M. O'Dell, *An Agent-Based Approach for Supporting Cross-Enterprise Workflows*, Proceeding of the 12th Australasian conference on Database technologies (2001), 123 – 130.