2007

Agent Based Supply Chain Architecture For Modern Supply Chain Management

Vilesh S. Salunkhe

Agent Based Supply Chain Architecture For Modern Supply Chain Management

Vilesh S. Salunkhe
B.E. (Electronics Engineering)
University of Mumbai, M.H.Saboo Siddik College of Engineering

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Department of Industrial and Systems Engineering in the Kate Gleason College of Enginerring of the Rochester Institute of Technology

October 2007

KATE GLEASON COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW TORK

CERTIFICATE OF APPROVAL

MASTER OF SCIENCE DEGREE THESIS

The M.S. Degree thesis of Vilesh S. Salunkhe has
been examined and approved by the thesis
committee as satisfactory for the thesis requirement
for the Master of Science degree.

Sudhakar Paidy

Dr. Sudhakar Paidy, Advisor

Moises Sudit

Dr. Moises Sudit

Thesis/Dissertation Author Permission Statement

Title of thesis or dissertation: Agent Based Supply Chain Architecture For Modern Supply Chain Management

Name of author: Vilesh S. Salunkhe
Degree: M.S.
Program:
College: Kate Gleason College of Engineering

I understand that I must submit a print copy of my thesis or dissertation to the RIT Archives, per current RIT guidelines for the completion of my degree. I hereby grant to the Rochester Institute of Technology and its agents the non-exclusive license to archive and make accessible my thesis or dissertation in whole or in part in all forms of media in perpetuity. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Print Reproduction Permission Granted:

I, _____, hereby **grant permission** to the Rochester Institute Technology to reproduce my print thesis or dissertation in whole or in part. Any reproduction will not be for commercial use or profit.

Signature of Author: _____ Date: _____

Print Reproduction Permission Denied:

I, Vilesh S. Salunkhe, hereby **deny permission** to the RIT Library of the Rochester Institute of Technology to reproduce my print thesis or dissertation in whole or in part.

Signature of Author: Vilesh S. Salunkhe Date: _____

Inclusion in the RIT Digital Media Library Electronic Thesis & Dissertation (ETD) Archive

I, _____, additionally grant to the Rochester Institute of Technology Digital Media Library (RIT DML) the non-exclusive license to archive and provide electronic access to my thesis or dissertation in whole or in part in all forms of media in perpetuity.

I understand that my work, in addition to its bibliographic record and abstract, will be available to the world-wide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I am aware that the Rochester Institute of Technology does not require registration of copyright for ETDs.

I hereby certify that, if appropriate, I have obtained and attached written permission statements from the owners of each third party copyrighted matter to be included in my thesis or dissertation. I certify that the version I submitted is the same as that approved by my committee.

Signature of Author: _____ Date: _____

Dedicated to

My Family

Acknowledgement

During the slow and very challenging process of development and implementation of this work I have accumulated many debts, only a portion of which I have space to acknowledge here.

I would like to thank Dr. Sudhakar Paidy for giving me the opportunity to do this work. It was under his guidance and direction that this work has been successfully completed. I want to thank him for supporting the idea of using agent technology for supply chain management and then introducing me to various different concepts like web services, XML etc. which were adopted for implementing this work. Dr. Paidy has been the most instrumental in making sure that this thesis makes a significant contribution to the academic literature. I would also like to thank Dr. Paidy for all the guidance and support he provided to me beyond the academic arena.

A special mention of thanks is owed to Dr. Moises Sudit, my thesis committee member for his helpful critique and encouragement.

I also want to thank, Dr. Jacqueline Mozrall and all other teaching members of the Industrial and Systems engineering department for supporting me during my stay.

I also want to thank Mrs. Marilyn Houck for providing me with all the administrative support and Mr. Tyler Rickettson for his assistance with computers and network related issues.

Many thanks are due to the computer science department, especially Dr. Schreiner and my friend Mr. Ratnakar Kamath with whose help I learnt the skills required for putting together an extensive .NET application, such as the one developed for this thesis.

I also want to thank Miss. Shalaka Deo for providing me with valuable help and support.

I want to thank all my friends for providing me their support in good times and bad.

Finally, I want to thank each and every person who has affected my work directly or indirectly and made this thesis's success possible.

ABSTRACT

A Supply chain is a linked series of various business partners that participate in the process of developing a finished product from raw materials. Supply chain management is the process of managing all the related activities. Though decisions are made by each business partner on their own, frequent interaction with other partners is a must. Timely response and high accuracy in the decisions made require collection of related data, analysis, and conclusion steps without missing a beat. However, these steps are tedious, complex, and very error prone as often these decisions are made by people with multiple responsibilities and frequent interruptions under the pressure to make timely decisions. The communication among the business partners also is very cumbersome and complex due to various application standards and hardware platforms used. Recent developments in information technology such as web services based applications have considerably improved communication among interacting partners in a supply chain. Web services provide a simple interoperable messaging framework in which data can be exchanged amongst different trading partners irrespective of their applications and platform. This work has successfully illustrated the process of automating some decision making processes in a supply chain by developing a representative supply chain (retailer, manufacturer, supplier and carrier) scenario using software agent technology and web services. Software agent (software module) is a decision making system which senses and acts in some environment. Autonomous software agents are agents which take decisions without any human intervention. They have properties of autonomy, reactivity, proactiveness and sociability. Software agent reduces human intervention and errors. The agents are based on a modified MASCOT agent architecture. These agents are implemented using web services and are used to automate routine business processes such as price negotiation, request for quote, earliest delivery date determination, purchase order cancellation etc. Open Applications Integration Specification (OAGIS) based standard Business Object Documents are used for communication amongst agents.

TABLE OF CONTENTS

List Of Figures

List of Tables

1. INTRODUCTION

"A supply chain is a network of facilities and distribution options that performs the functions of procurement of materials, transformation of these materials into intermediate and finished products and the distribution of these finished products to customers. Supply chain management, (SCM) is the systematic, strategic coordination of the traditional business functions and the tactics across these business functions within a particular company and across businesses within the supply chain, for the purpose of improving the long term performance of the individual companies and the supply chain as a whole. SCM requires the companies to make decisions regarding production, inventory, location, transportation and information, which are beneficial to all the concerned companies as a whole and then for the individual companies (Hugos 2003).

Communication with business partners is an important factor in making the supply chain efficient. Over time, new and advanced technologies have been introduced to improve communication and transparency of operations in a Supply Chain. Communication technologies like RPC, EDI, CORBA were used for B2B communication. These technologies considerably reduced the overhead that was present in information exchange among business partners. However, there was no consideration for interoperability among different technologies. If one business partner was using one particular technology, the other partners had to use the same technology for any communication to be possible, thus causing heavy IT infrastructure investment and partner lock-in. The evolution towards achieving interoperability amongst different platforms is achieved by using Web Services. Web Services technology is a collection of protocols and standards, developed specifically with interoperability in mind. Web Services enable the introduction of supply chain network architecture in which it is possible to add any number of suppliers, customers and logistics providers, form on-demand links with any other business partners and carry out supply chain transactions (Chandratre and Paidy, 2006)

Although Web Services eliminate the problem of interoperability, the supply chain network architecture depends on many manual interactions for decision making. The decision making

process relies heavily on information acquired manually and/or from external sources. Different decisions like deciding whether to initiate negotiation with customer, determining the earliest possible delivery date, calculating the penalty for cancelling a purchase order etc. Consider for example, if a supplier wants to plan his production based on the currently inventory level of his customers, he will have to constantly keep a check on the customers inventory level. When the inventory falls below a certain limit, the supplier can anticipate an order and plan his production accordingly. Doing this enables the supplier to be responsive to customer demands. This process requires a considerable human intervention in checking for the pre-determined inventory re-order level, generating and releasing a production order. On similar lines one can anticipate the amount of work required to perform various supply chain operations. Manually performing the tasks of placing a request for quote, selecting a supplier, carrying out price negotiations, placing a purchase order, offering discounts to eligible customers, selecting a carrier who can transport goods in a cheap, efficient and timely manner etc are tedious, time consuming and error prone. is a tedious, cumbersome and error prone activity.

Software Agent technology can be used to reduce or if possible completely eliminate the need of human intervention for decision making. An intelligent agent is an autonomous decision making system, which senses and acts in some environment (Ghiyoung, 2005). The technology of intelligent agents and multi agent systems can be used to conceptualize and implement complex, distributed supply chain architecture at the tactical and operational levels. It views the supply chain as composed of a set of intelligent (software) agents, each responsible for one or more activities in the supply chain and each interacting with other agents in planning and executing their activities (Wooldridge, 1997).

This thesis develops a architecture that uses agent technology based on MASCOT agent architecture and web services to create a supply chain scenario, in which some of the above mentioned tasks are automated (Sadeh, et al. 1999) (Sadeh et al, 1999). It creates an agent driven supply chain architecture and in the process partially automates some decision making tasks. MASCOT architecture is a re configurable, multilevel, agent based architecture for coordinated supply chain planning and scheduling designed to work in today's high-mix

production environment and works to improve coordination across multiple facilities and to evaluate new product/subcomponent designs and business decisions.

The following chapters of this document will examine and elaborate each of the facts mentioned above and each of the claims made.

Chapter 2 summarizes what research has already been done in this arena and what this thesis will contribute. Chapter 3 explains some basic principles of supply chain. It presents the concepts of horizontal and vertical collaboration. Chapter 4 explains web services and the technical details associated with web services. Chapter 5 provides an introduction to agent technology and gives some agent applications in supply chain management. It also provides some practical agent applications in industry. Chapter 6 explains MASCOT agent architecture. In chapter 7, the actual proposed architecture is described. The development of databases for the manufacturer, supplier and carrier using Microsoft SQL server 2005 Express is presented in chapter 8. Chapter 9 describes the implementation details and finally, chapter 10 presents the conclusions reached in this thesis.

Reference:

Hugos Michael. "Essentials of Supply Chain Management".

Chandratre Pradip and Paidy Sudhakar. "Web Services Infrastructure for Supply Chain", Proceedings of Sixteenth International Conference on Flexible Automation and Integrated Manufacturing, 2006.

Ghiyoung Im, Kun Chang Lee and Ohbyung Kwon. "MACE-SCM: An Effective Supply Chain Decision Making Approach based on Multi-agent and Case-based Reasoning". Proceedings of the 38th Hawaii International Conference on System Sciences, 2005.

M. Wooldridge. "Agent Based Software Engineering". IEEE online proceeding, 1997.
Norman M. Sadeh, David W. Hildum, Dag Kjenstad,and Allen Tseng. "MASCOT: An Agent-Based Architecture for Coordinated Mixed-Initiative Supply Chain Planning and Scheduling". Third International Conference on Autonomous Agents (Agents '99) Workshop on Agent-Based Decision Support for Managing the Internet-Enabled Supply Chain, Seattle WA, 1 May 1999.

Norman M. Sadeh, David W. Hildum, Dag Kjenstad, and Allen Tseng. "MASCOT: An Agent-Based Architecture for Coordinated Mixed-Initiative Supply Chain Planning and Scheduling". Third International Conference on Autonomous Agents (Agents '99) Workshop on Agent-Based Decision Support for Managing the Internet-Enabled Supply Chain, Seattle WA, 1 May 1999.

2. LITERATURE REVIEW

How successfully an organization is able to survive and prosper in its business environment depends in a large part on how quickly, efficiently and extensively it is able to create communication networks with its business partners. Trends of success and failure within businesses indicate that forming networks with a company's business partners is vital for the success of the company (Hengst, 2001). Use of technology enables a company to eliminate the problems associated with its operations.

To eliminate the tight coupling present during the use of technologies like RPC, EDI, CORBA and Java RMI, Web Service architecture is used. Web Services are self contained, modular business process applications based on the industry standard technologies of WSDL (to describe), UDDI (to advertise and syndicate), and SOAP (to communicate). They enable loose coupling by allowing users to connect different components across organizational boundaries in a platform- and language –independent manner (Leymann, 2004). Web services architecture is the dynamic creation of loosely coupled systems based on services. No single implementation technique is suggested (IBM Services Architecture team, 2000). The implementation of web services to demonstrate the ease of coupling between two business entities and the ease of adding new business entity with a specific role to the supply chain architecture was recently demonstrated (Chandratre, 2005; Chandratre and Paidy, 2006).

The technology of software based intelligent agents and multi agent systems can be used to conceptualize and implement complex, distributed supply chain architectures. Participating members of a supply chain make use of information system like enterprise resource planning (ERP) for planning and scheduling activities independently (Chan and Chan, 2004). Recent research indicates that there is a need to handle such distributed activities in an integrated manner, especially under fast and changing conditions. A multi agent system, a branch of distributed artificial intelligence, is a contemporary modeling technique for distributed systems (Chan and Chan, 2004). Participating members of a supply chain make use of information system like enterprise resource planning (ERP) for planning and scheduling activities independently. Recent research indicates that there is a need to handle such

distributed activities in an integrated manner, especially under fast and changing conditions. A multi-agent system, a branch of distributed artificial intelligence, is a contemporary modeling technique for distributed systems (Chan and Chan). An intelligent agent is an autonomous decision-making system, which senses and acts in some environment (Wooldridge, 1997). An *agent* is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives (Weiss, 1999). Software Agents are characterized by properties like autonomy, reactivity, pro activeness, sociability.

Agent technology can be used to reduce or completely eliminate human interaction in decision making. Software agents are capable of performing the tedious and error prone task of calculation based decision making. This can save organization valuable time and costs by avoiding errors.

Currently multi agent systems are conceptualized to solve a wide array of questions in supply chain management. Multi agent and "Case based reasoning" is suggested to solve complex problems such as Revenue maximization and stochastic dimensioning (Im, et al, 2005). Coordination scheme in Agile Supply Chain is explained using self interested and co operated agents (Ping Lou, 2005). The problem of non availability of necessary data throughout the supply chain prevents use of an integrated approach to planning. A solution to this problem is the implementation of a multi agent systems architecture using different systems each performing a specific function (Frey, et al, 2003). Genetic algorithms technique can be implemented to solve optimization problems (Yung and Yang, 1999).

Different agents have been developed to participate in the trading agent competition (TAC). The first TAC was held from June 22nd to July 8th 2000. The TAC was designed to create a benchmark problem in the complex domain of e-marketplaces (Stone, et al. 2001). TAC has different rules for every competition and based on these rules, designers have to model their agents to meet the competition requirements. Some examples of agents designed for TAC are ATTac-2000 which uses a principled bidding strategy that includes several elements of

adaptivity (Stone, et al, 2001). Mertacor agent was first in the classic TAC of 2005 (Toulis, et al, 2006). Many such examples are available.

"Multi Agent Supply Chain Coordination Tool" MASCOT, is one more example of multi agent system. MASCOT architecture is a reconfigurable, multilevel, agent based architecture for coordinated supply chain planning and scheduling designed to work in today's high-mix production environment and works to improve coordination across multiple facilities and to evaluate new product/subcomponent designs and business decisions. It provides a blackboard architecture which emphasizes modular encapsulation of problem solving knowledge within independent knowledge sources (Sadeh, et al, 1999).

A business process is a collection of inter related work tasks, initiated in response to an event that achieves a specific result for the customer. A large number of composition languages have been created for wiring together business processes. Business Process Execution Language for web services (BPEL4WS) is one such composition based workflow language for web services (Khalaf, et al, 2003). BPEL4WS, WS-Transaction, and WS-Coordination specifications are mainly used to orchestrate Web Services based application honoring the goals of service oriented paradigms of loose coupling, on demand interaction, quick adaptation to frequent change, and lack of control over the platform and implementation of services being used (Khalaf, et al, 2003).

To demonstrate the application of BPEL, consider a process in which a manufacturer places an order for raw materials with a supplier. The supplier then generates a cost estimate for the total order, which may also include the transportation cost of goods by a third party logistics provider. The manufacturer then can either accept this cost estimate or reject it. This process can be choreographed using BPEL by wiring together 3 web services, manufacturer's web service, logistic provider's web service and supplier's web service. Suppliers Web Service includes the logic to generate the cost estimate (Khalaf, et al, 2003).

Web service implementation solves the problem of tight coupling but its implementation in SCM still does not eliminate the human intervention required to make crucial decisions,

which may result in significant losses for the organizations. Also the work conducted in the area of Multi agent systems is heavily concentrated on the research side and very little work is done on implementing multi agent systems. The proposed study will attempt to incorporate agent technology with web services and provide a structure to demonstrate the automation of the decision making process in supply chain management and illustrate the practicality of using agents in supply chain to save time and avoid errors.

Reference:

"Web Services Infrastructure for Supply Chain", Master's thesis, Rochester Institute of Technology, 2005. Rochester, NY.

Christopher C. Yang and Stanley K. Yung. "Intelligent Multi-Agents for Supply Chain Management". IEEE 1999.

Frey Daniel, Peer-Oliver Woelk, Roland Zimmermann and Tim Stockheim. "Integrated Multi-Agent-Based Supply Chain Management". Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03).

Hengst Mariëlle den and Henk G. Sol. "The Impact of Information and Communication Technology on Interorganizational Coordination". Proceedings of the 34th Hawaii International Conference on System Sciences, 2001.

Hugos Michael. "Essentials of Supply Chain Management".

Im Ghiyoung, Kun Chang Lee and Ohbyung Kwon. "MACE-SCM: An Effective Supply Chain Decision Making Approach based on Multi-agent and Case-based Reasoning". Proceedings of the 38th Hawaii International Conference on System Sciences, 2005.

IBM Services Architecture team (Sep, 2000). "Web Services architecture overview The next stage of evolution for e-business". Retrieved March 26, 2006 from <http://www-128.ibm.com/developerworks/webservices/library/w-ovr/?dwzone=webservices#h2>.

Khalaf Rania, Mukhi Nirmal and Weerawarana Sanjiva. "Service-Oriented Composition in BPEL4WS". Proceedings of the twelfth International World Wide Web Conference, 20-24 May 2003.

Leymann Frank and Roller Dieter (Aug, 2004). "Business processes in a Web services world – A quick overview of BPEL4WS". Retrieved March 27, 2006 from <http://www-128.ibm.com/developerworks/library/ws-bpelwp/>.

Ping Lou, You-Ping Chen and Zu-De Zhou. "Study on Coordination in Multi-Agent-Based Agile Supply Chain Management". Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005.

Weiss, Gerhard. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, 1999.
Weiss, Gerhard. Multi agent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, 1999.

Wooldridge M. "Agent Based Software Engineering". IEEE online proceeding, 1997.

Chan, F T S and H K Chan. "A new model for manufacturing supply chain networks: a multi agent approach." Proc. Instn Mech. Engrs: Volume 218, Number 4. 1 April 2004. pp. 443-454(12).

Stone, Peter, et al. "Attac-2000: An Adaptive Autonomous Bidding Agent." AGENTS'01. Montreal, Quebec, Canada, May 28-June 1, 2001.

Toulis, Panos, Dionisis Kehagias and Pericles A. Mitikas. "Mertacor: A Successful Autonomous Trading Agent." AAMAS'06. Hakodate, Hokkaido, Japan.

Norman M. Sadeh, David W. Hildum, Dag Kjenstad,and Allen Tseng. "MASCOT: An Agent-Based Architecture for Coordinated Mixed-Initiative Supply Chain Planning and Scheduling". Third International Conference on Autonomous Agents (Agents '99) Workshop on Agent-Based Decision Support for Managing the Internet-Enabled Supply Chain, Seattle WA, 1 May 1999.

3. SUPPLY CHAIN ARCHITECTURE

A Supply Chain is a network of facilities and distribution options that performs the task function of procurement of materials, transformation of these materials into intermediate and finished products and the distribution of these finished products to the consumer (Ganeshan, 1995).

Supply chain management is the act of optimizing all activities throughout the supply chain, so that products and services are supplied to the consumers in the right quantity, to the right location, at the right time and at optimal cost (Clarkston, 2000). Effective management must take into account coordinating all the different pieces of this chain as quickly as possible without losing any of the quality or customer satisfaction, while still keeping costs down. It requires taking important decisions related to inventory, production, transportation, location and information, also known as "Five Major Supply Chain Drivers". This chapter explains the supply chain concept and its management, highlighting the importance of decision making process.

3.1. SUPPLY CHAIN

A supply chain encompasses the companies and business activities needed to design, make, deliver and use a product or service (Hugos, 2003). Business organizations are dependent on their supply chains to provide them with the raw materials or services required for their functioning. Some definitions of supply chain are offered below:

- "A supply chain is the alignment of firms that bring products or services to the market" (Lambert, 1998).
- "A supply chain consists of all stages involved, directly or indirectly. In fulfilling a customer request. The supply chain not only includes the manufacturers and suppliers, but also includes the transporters, warehouses, retailers and customers (Chopra, 2001)
- "A supply chain is a network of facilities and distribution options that performs the functions of procurement of materials, transformation of these materials into intermediate

and finished products, and the distribution of these finished products to customers." (Ganeshan, 1995)

Figure 3.1 below, shows the basic elements involved in a supply chain.

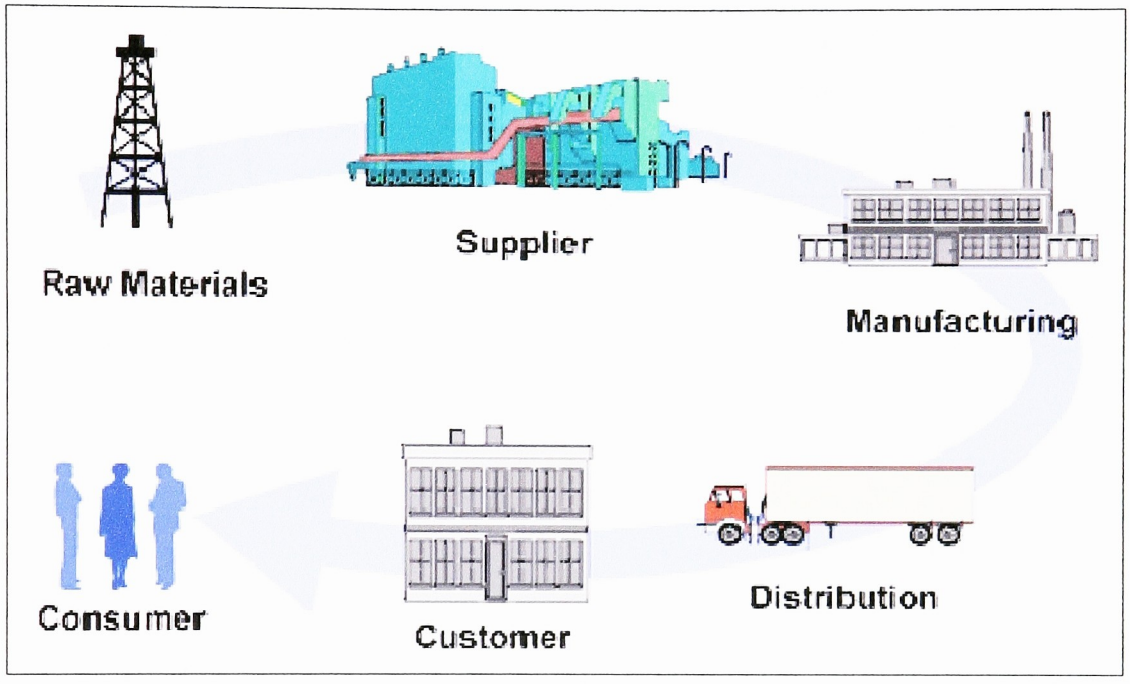


Figure 3.1: Basic Elements of a Supply Chain (The Progress Group, 2007)

In its simplest form, a supply chain consists of a company, its suppliers and its customers, as shown in figure 3.2. An extended supply chain will additionally have a supplier's supplier (ultimate supplier) and, similarly an ultimate customer. Additionally, there are companies which function as service providers to the companies in a supply chain. These are companies that provide services in logistics, finance, marketing and information technology (Hugos, 2003). Figure 3.3 shows an extended supply chain.

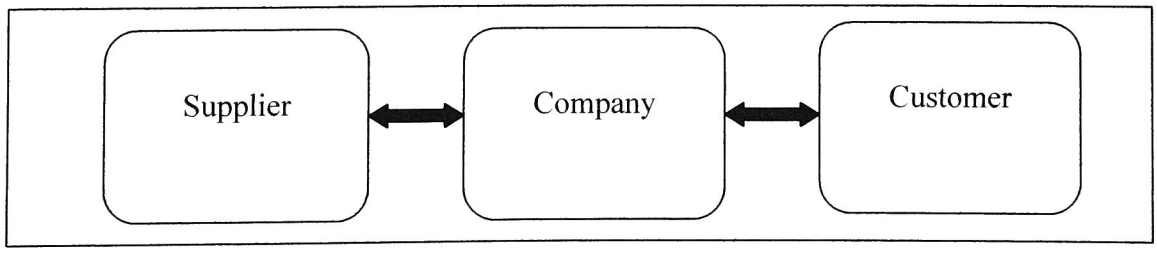


Figure 3.2: A Basic Supply Chain (Hugos, 2003)

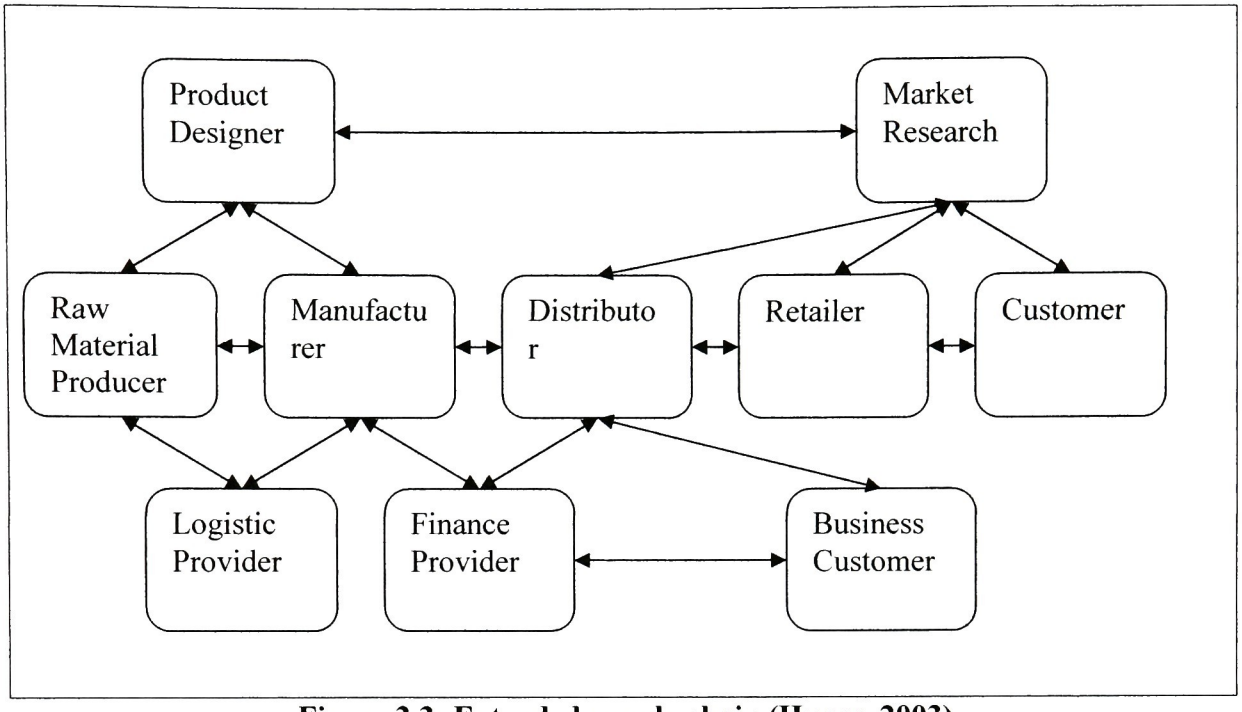


Figure 3.3: Extended supply chain (Hugos, 2003)

Following section provides a brief overview of the various supply chain participants.

Producers

Producers or manufacturers or suppliers are companies responsible for making products. The products can be raw materials or finished goods. Making a product often requires the producer to gather raw materials from his ultimate producer. He has to decide what, when and how much of raw materials he is going to require for producing his product, in time, for his consumer.

Distributors

Distributors or wholesalers are intermediaries between manufacturer and retailer. Distributors take inventory in bulk from producers and generally sell to other businesses. They are required to perform different functions regarding product promotion, sales, tracking of customer requirements, management of inventory etc.

Retailers

Retailers stock inventory and sell it directly to the retail customer. They keep a close watch on the preferences and demands of the customers. It advertises and tries to attract customers by using seasonal promotions, low prices and/or product selection, services, convenience, etc.

Customer

A customer or consumer is any individual or organization that purchases and uses a product.

Service Providers

These are organizations that provide services to producers, distributors, retailers and customers. Service providers have the necessary expertise to perform certain functions that are needed in a supply chain of any kind of product or service. Instead of each producer investing in managing such functions themselves, solution providers can do it more effectively and economically. Common service providers are providers of transportation and warehousing.

Supply chain concept has evolved from a simple structure in the old industrial age to its current complex form (Hugos, 2003). Traditionally, to exploit benefits of large scale production, companies internally performed most, if not all, the operations required to deliver the end product to the consumer: manufacture their own raw materials, product manufacturing, transportation services etc. This structure, even though, provided the companies with benefits in large scale production; it afforded very little flexibility in terms of product design. Over time, the consumers have developed a wider appetite for product design because of which the manufacturer has very little time to deliver the product to the market. This requirement makes the vertical structure of companies obsolete. To overcome this challenge, of quick delivery to the market, companies have shifted from their traditional integrated structure towards a horizontal structure, comprising of many companies, each performing a specific activity in the supply chain. Figure 3.4 helps to visualize this transformation.

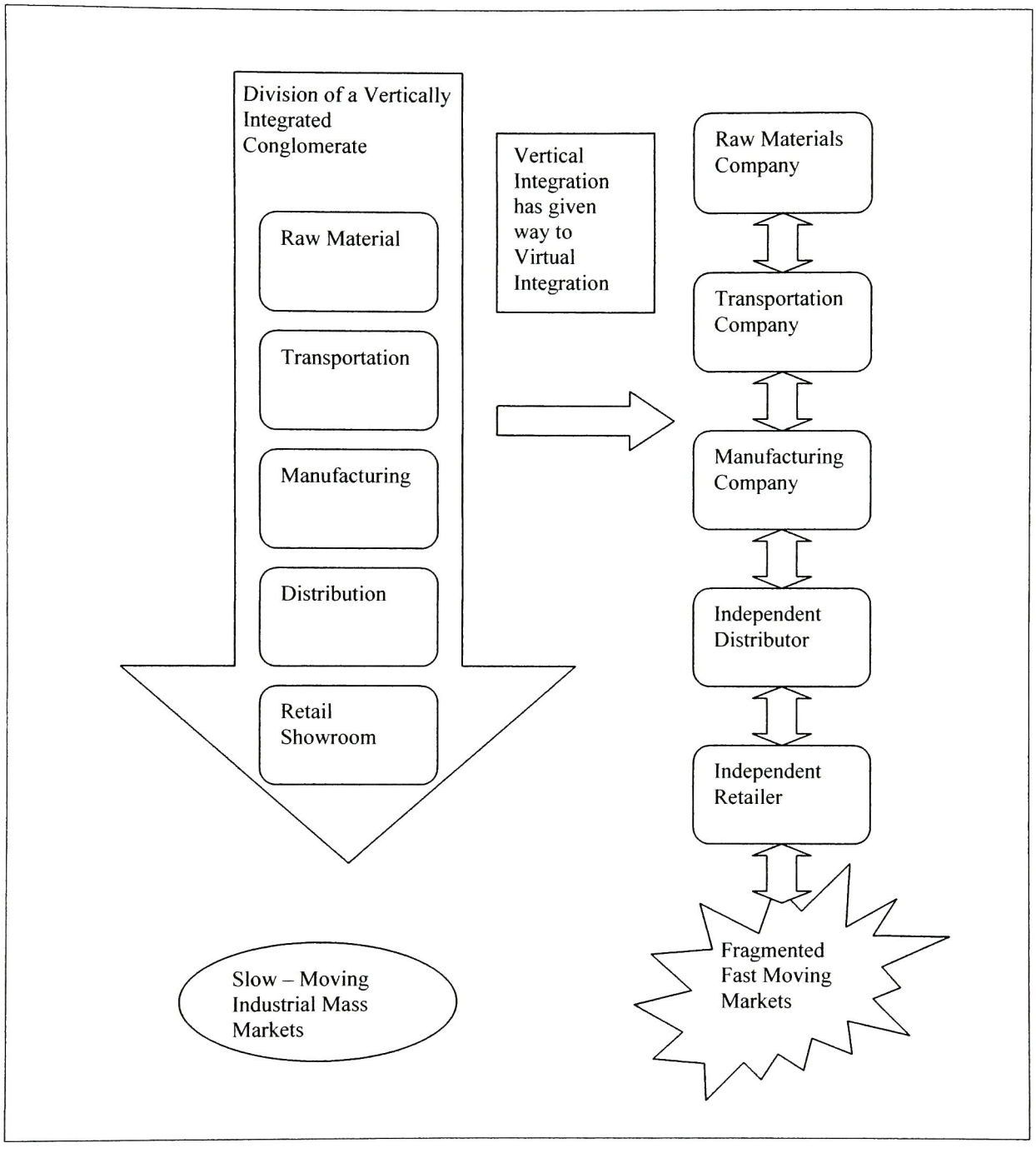


Figure 3.4: Evolution from a vertically integrated supply chain to a "virtually integrated" supply chain (Clarkston, 2000)

3.2. SUPPLY CHAIN MANAGEMENT

Different supply chain partners interact with each other, for either material or information exchange. This interaction has to be coordinated for the success of an organization and also the supply chain as a whole. More formal definitions of supply chain management are as given below:

- "Supply chain management is the process of planning, implementing and controlling the operations of a supply chain with the purpose to satisfy the customer requirements as efficiently as possible. Supply chain management spans all movement and storage of raw materials, work in process inventory, and finished goods from point of origin to point of consumption" (Wikipedia, "Supply Chain Management", 2007).
- "The systemic, strategic coordination of the traditional business functions and the tactics across these business functions within a particular company and across businesses within the supply chain, for the purposes of improving the long-term performance of the individual companies and the supply chain as a whole." (Mentzer, 2001).
- "Supply chain management is the coordination of production, inventory, location, and transportation among the participants in a supply chain to achieve the best mix of responsiveness and efficiency for the market being served" (Hugos, 2003).
- Supply chain management involves making strategic and operational decisions. Strategic decisions are made over a longer time horizon and are closely related to corporate strategy. On the other hand, operational decisions are short term, and focus on day-to-day basis (Ganeshan, 1995)

Companies which are part of a supply chain need to collaborate with each. Collaboration is a very broad and encompassing term. Many authors when talking about collaboration cite mutuality of benefit, rewards and risk sharing together with the exchange of information as the foundation of collaboration (Stank, 1999). A point of interest here would be to look at different types of collaborations in a supply chain. Figure 3.5, shows vertical and horizontal collaboration. In vertical collaboration, an organization collaborates with its customers, internally across its functions and with its suppliers e.g. improving material forecasts. In

horizontal collaboration, organizations can collaborate with competitors, internally and with non-competitors e.g. sharing manufacturing capacity.

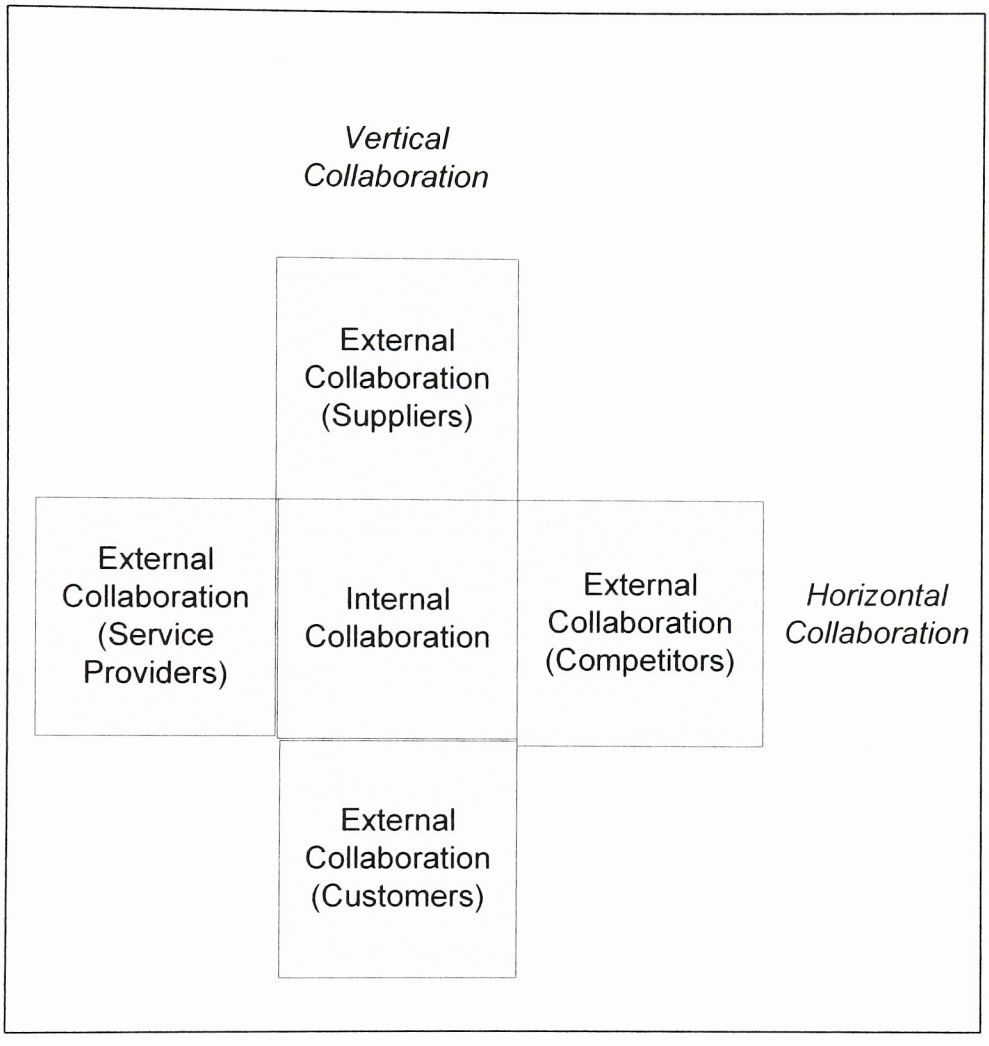


Figure 3.5: Scope of Collaboration (Baratt, 2004)

There are four major decision areas in supply chain management (Ganeshan, 1995):

- Location Decisions

  Location refers to the geographical sitting of the supply chain facilities. It includes the decisions related to which activities should be performed in each facility. Here the companies need to decide whether to centralize activities in fewer locations to gain economies of scale and efficiency or to decentralize activities in many locations close to

customers and suppliers in order for operations to be more responsive. These decisions are of great significance to a firm since they represent the basic strategy for accessing customer markets, and will have a considerable impact on revenue, cost, and level of service.

- Production decisions

  Production is a module in a supply chain which has a capacity to make and store products. The strategic decisions include what products to produce, which plants to produce them in, allocation of suppliers to plants, plants to distribution centers and distribution centers to customer markets. Operational decisions focus on detailed production scheduling. These decisions include the construction of the master production schedules, scheduling production on machines, and equipment maintenance. Other considerations include workload balancing, and quality control measures at a production facility.

- Inventory Decisions

  Inventory is a non value added module, which is spread throughout the supply chain and includes everything from raw material to work in progress to finished goods that are held by manufacturers, distributors and retailers for easy accessibility in a supply chain. Inventory decisions refer to means by which inventories are managed. Their primary purpose is to buffer against any uncertainty that might exist in the supply chain. Since holding of inventories can cost anywhere between 20 to 40 percent of their value, their efficient management is critical in supply chain operations. The operational strategies include deployment strategies (push versus pulls), control policies --- the determination of the optimal levels of order quantities and reorder points, and setting safety stock levels, at each stocking location. These levels are critical, since they are primary determinants of customer service levels.

- Transportation Decisions

  This refers to the movement of everything from raw material to finished goods between different facilities in a supply chain. The mode choice aspects of these

decisions are more strategic and are closely linked to the inventory decisions. Air shipments are fast, reliable, and warrant lesser safety stocks but they are expensive. While shipping by sea or rail is much cheaper, they necessitate holding relatively large amounts of inventory to buffer against the inherent uncertainty associated with them. Therefore customer service levels and geographic location play vital roles in such decisions.

As an example of collaboration between two organizations in a supply chain, consider the collaboration between UPS and Harley-Davidson (UPS Supply Chain Solutions, 2007). Internal forecasts for Harley-Davidson expected an increase of sales. This expected increase posed a challenge to the company's logistic network, particularly for inbound transportation and supply of parts and accessories to dealerships. In order to focus on its competency, Harley-Davidson sought help from UPS logistics. To solve the companies' logistic problem of frequent less than full truck deliveries of shipments from supplier, UPS directed 16 of the companies' suppliers supplying directly to Harleys Wisconsin plant, to direct their deliveries to its Chicago based cross docking facility. Using historic data from the suppliers, the analysts calculated optimal shipping frequencies for each of the suppliers, enabling the company to reduce the rate of regular inbound shipments to three times a week, while also giving the manufacturing plant faster access to larger stocks of parts.

All the decision making requires significant intervention from people with multiple responsibilities and people who are under time constraints. Use of agent technologies, can help to reduce or eliminate human interaction and errors can be avoided.

Reference:

Ganeshan, Ram, and Terry P. Harrison, 1995,"An Introduction to Supply Chain management," Department of Management Sciences and Information Systems, 303 Beam Business Building, Penn State University, University Park, PA .1995.

Clarkston. Supply Chain Management Primer. 2000.

Hugos, M. Essentials of Supply Chain Management. John Wiley and Sons, Inc, 2003.

Lambert, Douglas M., James R. Stock, and Lisa M. Ellram, 1998, "Fundamentals of Logistics Management", Boston, MA: Irwin/McGraw-Hill, Chapter 14

Chopra, V. Sunil, and Peter Meindl, 2001, "Supply Chain Management:Strategy, Planning, and Operations", Upper Saddle River, NJ: Prentice-Hall, Inc. Chapter 1

"The Progress Group", Retreived March 2007 from http://www.theprogressgroup.com/publications/wp_images/fite1.gif,

Wikipedia. "Supply Chain Management" Retreived June 24, 2007 from http://en.wikipedia.org/wiki/Supply_chain_management

Mentzer, John T. ,William DeWitt, James S. Keebler, Soonhong Min, Nancy W. Nix, Carlo D. Smith, and Zach G. Zacharia, 2001, "Defining Supply Chain Management," Journal of Business Logistics, Volume 22, No. 2, p. 18

Stank, T.P., Crum, M. and Arango, M. (1999), ``Benefits of inter-firm co-ordination in food industry supply chains'', Journal of Business Logistics, Vol. 20 No. 2, pp. 21-41.

Mark Baratt, 2004, "Understanding the meaning of collaboration in the supply chain", Supply Chain Management: An International Journal, Volume 9. Number 1, pp. 30 – 42.

UPS Supply Chain Solutions. "Harley-Davidson Revs Up Logistics to Keep Pace With Growth" Retreived on July 7, 2007 from http://www.ups-scs.com/solutions/case_studies/cs_harley.pdf, 2007, 07-06-07.

Weiss, Gerhard. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, 1999.

Weiss, Gerhard. Multi agent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, 1999.

Chan, F T S and H K Chan. "A new model for manufacturing supply chain networks: a multiagent approach." Proc. Instn Mech. Engrs (2004): 443-454.

4. WEB SERVICE ARCHITECTURE

4.1. INTRODUCTION TO WEB SERVICES

The World Wide Web Consortium defines a web service as a software system designed to support interoperable machine to machine interaction over a network (http://www.w3.org/TR/ws-arch/#whatis, 2006).

Web Services are software programs which have the capacity to interact with any other software programs, regardless of the language in which the two programs are written, as long as the two software programs adopt the same standards of communication. The idea of Web Services is to allow communication between two computer programs without any human intervention. Web Services are self contained, self-describing, modular applications that can be published, located and invoked across the web (IBM Glossary).

Web services have an interface described in a machine processable format i.e. Web Service Description Language (WSDL). Other systems interact with a web service in a manner prescribed by its definition using Standard Object Access Protocol (SOAP) messages, typically conveyed using HTTP with an XML serialization in conjunction with other web related standards (http://www.w3.org/TR/ws-arch/#whatis, 2006).

The web services can be used to develop complex software systems. With the evolution of the web technologies, companies developed software customized to their specific needs. However, with mergers, acquisitions and business growths, companies needed to share information. This required a huge investment on part of the company to modify the software systems to enable information sharing. Different technologies like RMI, COM, CORBA, EDI, ebXML have the ability to link different software systems, however it requires a steep learning curve to implement them (Lakshmi Ananthamurthy, 2007). As web services hide the internal software infrastructure for the outside world and use XML as a base language over SOAP for communication, they lend themselves as a very user friendly technology. Web services do not impose any compulsion to use any specific software. They modularize the business processes. Each web service represents a particular function. A service focuses on

what is being done, not how. By thinking in terms of services, the focus is on the business function, not the underlying technology (Connell, par. 17). Web services enable loose coupling between a service provider and a service requestor. So in an ideal web services world, a service requestor can change his/her service provider without much of investment if he/she does not like the service provider's quality of service (Chandratre, 2005).

Within a single enterprise itself, instead of having one big enterprise wide software, different components built by different vendors could be mixed and matched to create an enterprise application unique to the enterprise without any of the development work.

Web services approach to building complex software systems bears many similarities to the engineering process of a collection of software agents. In particular, large systems are assembled from distributed heterogeneous software components providing specialized services and communicating using agreed upon protocols. The area of Web Services offers much of real interest to the Multi Agent community, with reference to system architectures, powerful tools and the focus on issues such as security and reliability (AAMAS, 2003.)

4.2. WEB SERVICES TECHNOLOGY

Web services are simple, self contained applications which perform functions, from simple requests to complex business processes (Australian Government Technical Interoperability Framework, Glossary, 2006). The web service software can be written in any language that is suitable. This software has to be able to communicate with other software applications (Chandratre, 2005). The web services model uses WSDL, UDDI and SOAP. A WSDL is retrieved from the UDDI directory. WSDL descriptions allow the software systems of one business to extend to use those of other directly. The services are invoked over the World Wide Web by using the SOAP protocol. Each of the components is XML based. When two agencies know about each other's web service's they can link their SOAP interfaces provided all security concerns are managed. However, when services have unknown user, they need to be formally described by the WSDL language and entered into the UDDI directory (Australian Government Technical Interoperability Framework, Glossary, 2006). Figure 4.1 shows a web service provider, service requester and its binding and publication in a UDDI

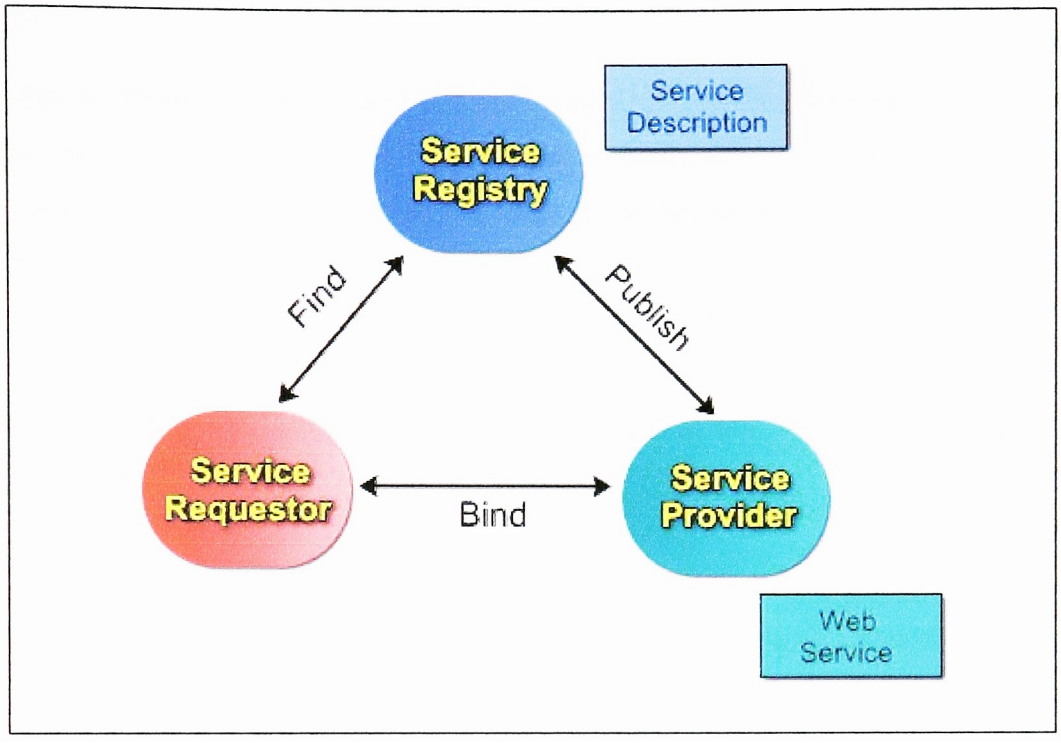


Figure 4.1: Web Service actors, objects and operations (Brittenham)

The IBM's idea of the Web Services conceptual stack is shown in figure 4.2

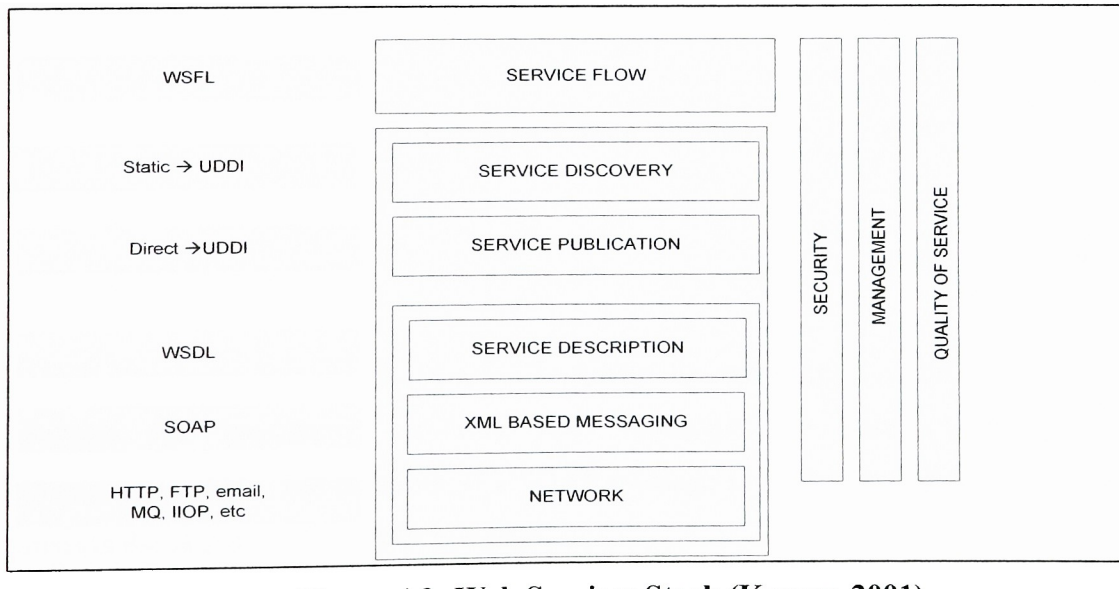


Figure 4.2: Web Services Stack (Kreger, 2001)

4.3. SOAP

Simple Object Access Protocol is a web protocol to exchange information from one computer to another. SOAP is a way for a program running in one kind of operating system to communicate with a program in the same or another kind of operating system by using the World Wide Web's Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML) as the mechanism for information exchange. SOAP specifies exactly how to encode an HTTP header and an XML file so that a program in one computer can call a program in another computer (Australian Government Technical Interoperability Framework, Glossary, 2006).

SOAP is basically an HTTP post with an XML envelop as a payload but with additional header information. SOAP is preferred over simple HTTP post of XML, because it defines a standard mechanism to incorporate orthogonal extensions to the message using SOAP headers and a standard encoding of operation or function (Kreger, 2001).

The W3C specification defines SOAP as follows: SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML – based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

4.3.1. SOAP MESSAGES

SOAP messaging framework uses XML syntax for message. It defines a suite of XML elements for "packaging" arbitrary XML messages for transport between systems (Chandratre, 2005). The structure of a SOAP message is as shown in figure 4.3. Figure 4.4 presents the skeleton of a SOAP message.

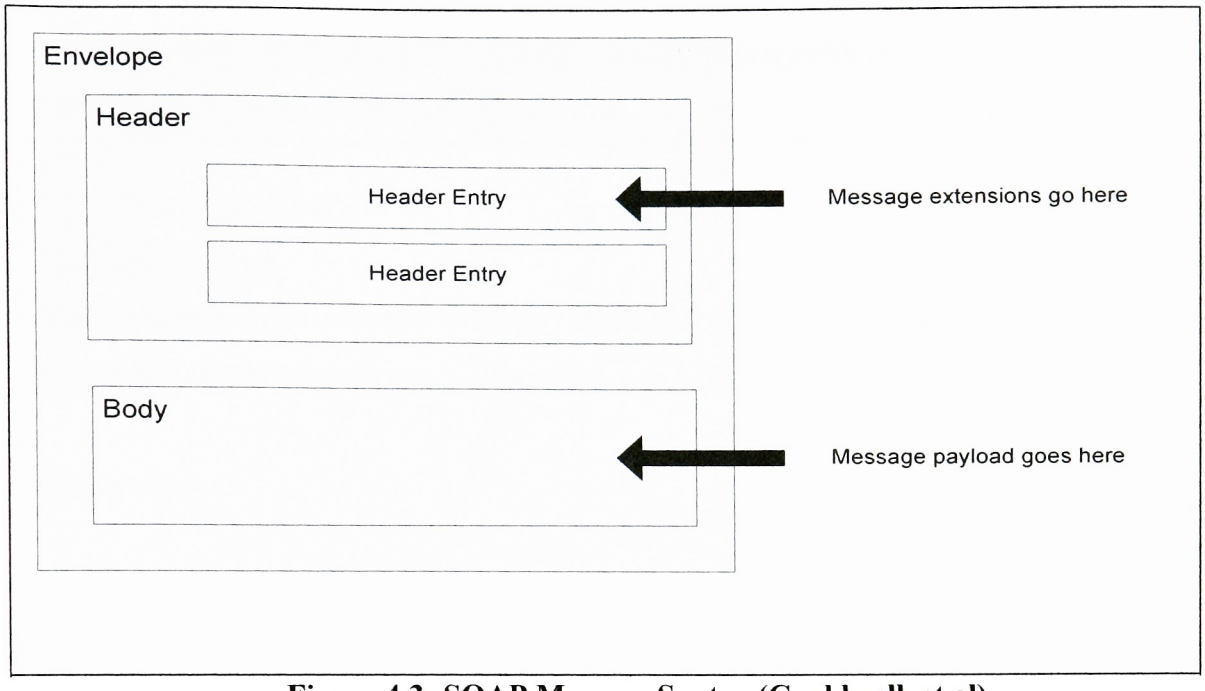


Figure 4.3: SOAP Message Syntax (Cauldwell, et al)

The building blocks of a SOAP message are constituted by the following (W3C, 2007):

- A required "Envelope" element that identifies the XML document as a SOAP message.
- An optional "Header" element that contains header information.
- A required "Body" element that contains call and response information.
- An optional "Fault" element that provides information about the errors that occurred while processing the message.

The required SOAP Envelope element is the root element of a SOAP message. It defines the XML document as a SOAP message. The optional SOAP Header element contains application specific information about the SOAP message. If the Header element is present it must be the first element after the envelope element. The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message. The optional SOAP Fault element is used to hold error and status information for a SOAP message (W3C, 2007).

SOAP makes extensive use of namespacing and attribute specification tags in almost every element of a message (Rhodes, 2007).

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  ...
  ...
</soap:Header>

<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
  ...
</soap:Body>

</soap:Envelope>
```

Figure 4.4: Skeleton SOAP message

4.4 WEB SERVICE DESCRIPTION, PUBLICATION AND DISCOVERY:

4.4.1 WSDL (WEB SERVICES DESCRIPTION LANGUAGE):

The technical details of a web service are provided in an XML document known as a WSDL document. WSDL stands for Web Services Description Language. WSDL is a machine processable document. WSDL describes a web service. It specifies the location of the service, the operations the service exposes, the input and output parameters, return values, and the data types used. The main structure of a WSDL document is as shown in figure 5 (W3C, 2007).

```
<definitions>

<types>
  definition of types.....
</types>
<message>
  definition of a message....
</message>
<portType>
  definition of a port.....
</portType>
<binding>
  definition of a binding....
</binding>

</definitions>
```

Figure 4.5: WSDL Document structure

A WSDL document has the following major elements (W3C, 2007):

<types>
The data types used in a web service are described here.
</types>

<message>
The messages used by the web service are described here.
</message>

<portType>
The portType element defines a web service, the operations that can be performed, and the messages that are involved. It defines the connection point to the web service.
</portType>

<binding>
WSDL bindings defines the message format and protocol details for a web service.
</binding>

A WSDL document can also contain other elements, like the extension elements and a service element that makes it possible to group together the definition of several web services into one single document.

4.5 UDDI (UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION)

Universal Description, Discovery and Integration (UDDI) is a directory service where businesses can register and search for Web Services. It is a platform independent framework for describing services, discovering businesses, and integrating business services by using the internet. UDDI is a directory for storing information about web service interfaces described using WSDL (W3schools).

Reference:

Web Services Architecture. Retrieved 2006 from http://www.w3.org/TR/ws-arch/#whatis

Lakshmi Ananthamurthy. Introduction to Web Services. Retrieved July 9, 2007 from http://www.developer.com/services/article.php/1485821

Chandratre Pradip. "Web Services Infrastructure for Supply Chain", Master's thesis, Rochester Institute of Technology, 2005. Rochester, NY.

Connell. "Web Services in Action: Aligning IT with Business Objectives". WebService.org. (2003 October) http://www.webservices.org/index.php/article/articleview/1152/1/7/

AAMAS 2003. Retreived on July 9, 2007 from http://agentus.com/WSABE2003/

"Australian Government Technical Interoperability Framework, Glossary". (2005) Retrieved on July 8, 2007 from http://www.agimo.gov.au/publications/2005/04/agtifv2/glossary

Kreger. " Web Service Conceptual Architecture". IBM.com (2001 May) http://www.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

Cauldwell, Patrick, Rajesh Chawla, Vivek Chopra, Gary Damachen, Chris Dix, Tony Hong, Francis Norton, Uche Ogbuji, Glen Olander, Mark A.Richman, Kirsty Saunders and Zoran Zaev. "Professional XML Web Services". Chapter 3: SOAP Basics.

W3C. "SOAP Syntax" Retrieved on July 9, 2007 from http://www.w3schools.com/soap/soap_syntax.asp (W3schools).

Kate Rhodes. " XML-RPC vs. SOAP". Retrieved on July 9, 2007 from http://weblog.masukomi.org/writings/xml-rpc_vs_soap.htm

Brittenham, Peter. "An overview of the Web Services Inspection Language." 01 June 2002.

IBM. July 2007 <http://www.ibm.com/developerworks/library/ws-wsilover/index.html>.

5. AGENTS IN SUPPLY CHAIN MANAGEMENT

5.1. INTRODUCTION

The supply chain is a worldwide network of supplier, factories, warehouses, distribution centers, and retailers through which raw materials are acquired, transformed, and delivered to customers. Businesses depend on their supply chains to provide them with what they need to survive and thrive. Supply chain management is the systematic, strategic, tactical, and operational decision making that optimizes supply chain performance (Wooldridge, 1997, Stuart et al, 1995). The strategic level is where top level management decisions are taken. Decisions related to supplier selection, transportation routes, manufacturing facilities etc., which relate to the long term planning of the organization. The tactical level, plans and schedules the supply chain to meet the actual demand (Wooldridge, 1997). The operational level executes the plans. Supply chain management requires companies to make decisions regarding production, inventory, transportation and information, which are beneficial to all the concerned companies as a whole and then for the individual companies.

Over time, the choices of the customers have changed and there is a lot of variation and uncertainty in their product demand. To maximize its profit the supply chain should be able to meet the demands of the customers and without any time lag. The use of latest technologies in information systems allows the organizations to, effectively respond to customer demands, to make production plans at the strategic and organizational level, manage the inventory at optimum level to avoid incurring loss due to excessive storage charge or lost orders, decide the transportation schedule etc. thus enabling them to respond efficiently to customer demand. Latest and advanced information technologies also help to solve all the strategic decision making problems in a supply chain. They can be used to select the appropriate location for a new plant, choose a suitable partner for the supply chain, form collaboration with new partners with minimum hassle etc.

The technology of intelligent agents and multi agent systems can be used to conceptualize and implement complex, distributed supply chain architecture at the tactical and operational

levels. It views the supply chain as composed of a set of intelligent (software) agents, each responsible for one or more activities in the supply chain and each interacting with other agents in planning and executing their activities (Wooldridge, 1997). An intelligent agent is an autonomous decision making system, which senses and acts in some environment (Ghiyoung Im et al, 2005) . It is an autonomous, goal oriented software process that operates asynchronously, communicating and coordinating with other agents as needed (Wooldridge, 1997). A multi agent system is one in which a number of heterogeneous intelligent agents, each having its own diverse goals and capabilities are employed to solve the various problems of a firm. For example, the DISPOWEB multi agents system can be individually employed by all the different organizations in a supply chain for the purpose of strategic level planning. Multi agents along with Case Based Reasoning can be used to solve different Optimization problems (Ping et al, 2005). Agent technology can be used to carry out co operative work by inter operation across networked humans, organizations and machines.

This section explains subjectively the structure of an agent entailing its different properties (Ghiyoung Im et al, 2005). It also explains how multi agents and CBR are used to perform optimization in a supply chain (Ping, 2007). Two different co ordination techniques are explained for two different types of agents (Frey et al, 2003). Then it also shows how different multi agents systems can be integrated to do production planning and control (Yang, 1999). How Genetic Algorithms are used to do optimization for warehouse management is also explained (Hugos, 2003). In the end, it gives some examples of different agents present in the industry.

5.2. AGENT TECHNOLOGY

An autonomous agent is a system situated within and a part of an environment it senses and acts on it, over time, in pursuit of its own agenda, so as to affect what it senses in the future. An environment may be the physical world, a user via a graphical user interface, a collection of other agents, the internet or a combination of these.

An agent based system is one in which the key abstraction used is that of an agent. Agent based systems may contain a single agent, e.g. user interface agents or software secretaries or they may be multi agent systems, e.g. DISPOWEB multi agent system (Wooldridge, 1997).
From the above definitions, one observes that an agent is a system with the following properties (Wooldridge, 1997).

- *Autonomy*

  Agents encapsulate some state (that is not accessible to other agents), and make decisions about what to do based on this state, without the direct intervention of humans or others.

- *Reactivity*

  Agents are situated in an environment and are able to perceive this environment, through the use of potentially imperfect sensors, and respond in a timely fashion to changes that occur in it.

- *Proactiveness*

  *A*gents do not simply act in an environment; they are able to exhibit goal directed behavior by taking the initiative.

- *Sociability*

  *A*gents interact with other agents (and possibly with humans) via some kind of agent communication language, and typically have the ability to engage in social activities to achieve their goals.

These properties can be understood more clearly with the help of the following example:
Consider an autonomous automatic pilot controlling an aircraft. The goal of autonomous pilot is to safely land at some airport. It is supposed to plan how to achieve this goal perhaps by making use of precompiled plans, rather than reasoning from first principles, and if necessary, to generate subsidiary goals e.g. ascend to an altitude of 30 000 feet, then proceed due north at a certain speed. This is what is meant by proactiveness. It is also expected from the system to try to execute its plans, but not blindly. That is in the event of an unforeseen circumstance e.g.

a change in weather conditions, a fault in the aircraft, a request from air-traffic control, the system should respond to the new situation accordingly, in time for the response to be useful. A system that spends hours deliberating about what to do next would be no use as an autopilot. This is what is meant by reactiveness. The autopilot is also expected to be able to cooperate with air-traffic controllers and perhaps other aircraft to achieve its goals. This is what it means by social ability

5. 3. APPLICATIONS OF AGENTS

Applications of agent technology can be classified into four areas (http://www.eil.utoronto.ca/profiles/rune/node6.html, 2006):

- *Interface Agent*

  Where an agent is an interface between the user and for example a source of information. The agent can co operate with the user in filtering out the information that is of interest to the user.

- *Information Agent*

  Could retrieve information from one or more sources of information upon request from the user.

- *Believable Agent*

  Is one who ``provide the illusion of life, thus permitting the audience's suspension of disbelief''

- *Cooperative Problem Solving and Distributed AI*

  This area has the largest potential for agent application. Current applications like power systems management, air-traffic, particle accelerator control, intelligent document retrieval, patient care, telecommunications network management, spacecraft control, computer integrated manufacturing, concurrent engineering, transportation management, job shop scheduling, and steel coil processing control involve cooperative problem solving agent. It is within this area that one finds large industrial applications.

5.4. AGENTS IN SUPPLY CHAIN MANAGEMENT

Using Multi agent systems in a supply chain allows us to easily conceptualize a system. This is most evident where the application domain is readily conceived in terms of naturally occurring entities, which is the case for a supply chain. A supply chain (as previously defined) can be visualized as a set of entities and processes. Entities may be suppliers, plants, distributions centers etc. or it may be internal departments such as sales, planning, purchasing, materials or research and development. A process is simply a series of actions. An entity is responsible for a set of processes, e.g. sales might be responsible for processes related to order acquisition, purchasing for processes related to supplier selection and material ordering, and R&D's responsibility is processes related to introduction of new products in the supply chain.
A supply chain is a domain which is frequently subjected to structural changes. Agents are autonomous, and often distributed, with very clearly defined interfaces, i.e. message passing. This gives a robust system that can undergo continuous adaption to the changes in the environment, both locally and globally, without the degradation of performance. Automated procedures can be developed to deal with adding and removing agents to the system, and changes within an agent will not affect other agents.

5.4.1. SUPPLY CHAIN DECISION MAKING BASED ON AGENTS (Im, et al, 2005)

The following section describes the research done in developing Multi Agent Collaboration Engine for supply chain model.

Case Based Reasoning (CBR) is a problem solving approach that relies on past cases to solve a particular problem. A CBR system finds its solution by searching its case base and comparing the attributes of each case to the problem to be solved. It then reads similar cases and derives the final answer by adjusting the differences between the current situation and the ones described in the previous cases. Multi-agent and case based reasoning are applied to facilitate collaboration and information sharing for solving production planning optimization problems. Multi-agent collaboration engine for supply chain management (MACE-SCM) is used to manage two decision support levels that reflect different types of relationships among the firms in a supply chain.Using a CBR based approach has the following advantages – It

allows us to solve problems without having to create solutions from scratch each time. CBR is flexible in handling the uncertainty of demand and supply and CBR also take advantage of a firm's knowledge

Consider the case of revenue maximization with regard to production and marketing decisions. Here instead of directly considering marketing model, marketing information is used to make planning decisions. In the MACE-SCM, two different levels of collaboration among the firms in a supply chain are developed: Autonomy Level and Collaboration Level, to model supply side uncertainties due to relationship changes among firms in a supply chain. At the Autonomy Level, firms in the supply chain maintain an arms-length relationship and do not collaborate to generate the solutions for the entire supply chain. At the Collaboration Level, firms maintain strategic relationships and collaborate extensively to generate the solutions for the entire supply chain. At this level, the firms may or may not rely on a case base.

This research considers a supply chain consisting of retailers, manufacturers and suppliers. A customer visits retailer and purchases the product if it is in stock. If the customer demand cannot be met by the retailer, it is ordered from a manufacturer and backlogged. The manufacturer produces products by assembling components from a supplier. The manufacturer receives product orders from the retailer and places component orders to the supplier. The supplier produces components and supplies them to the manufacturer. The supply chain incurs linear holding costs and linear backorder costs at each stage. The goal of the retailer is to maximize profits by minimizing stock outs and inventory costs. The manufacturer pursues profit maximization through the minimization of inventory costs and efficient management of manufacturing and procurement processes. The supplier is interested in maximizing profits by maintaining low turnaround times and low inventory.
The overall framework of the MACE-SCM is as shown below:

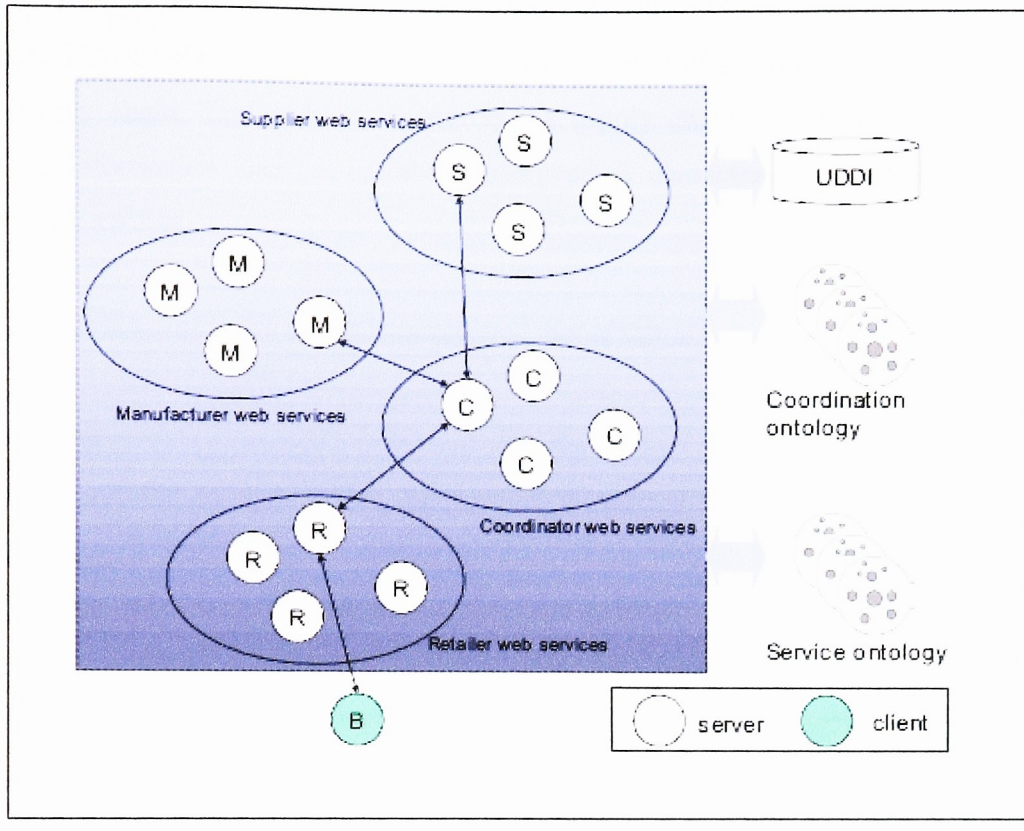


Figure 5.1: MACE SCM system framework ***(Im, et al, 2005)***

There are three distinct components in the framework: agent-based web services, coordination and service ontology's, and the web service directory (UDDI). The retailer agent (R-Agent), the manufacturer agent (M-Agent), and the supplier agent (S-Agent) model the retailer, the manufacturer and the supplier, respectively. A coordinator agent for the supply chain is also introduced. It manages more strategic collaboration amongst agents. Here, a buyer initiates a transaction by visiting an R-Agent. To find out an appropriate R-Agent, the buyer's program may visit the service repository (UDDI). This repository is used to search the Uniform Resource Identifier (URI) address that points to an agent's location and to acquire the address of the service ontology as URI. The buyer's program then imports the R-Agent's detailed data such as products, contact information, advertisements, and price structures. Such data is used to make a decision to select the best retailer. The buyer's program is the client module that can invoke any retailer's web services in the supply chain. The selected R-Agent then seeks a coordinator web service by visiting the collaboration ontology. The collaboration ontology

holds collaboration information required during the collaboration processes between the web services. The coordination and service ontology's consist of a common language agreed upon by a supply chain. The software agents in the supply chain use the ontology to exchange semantic information such as collaboration, product characteristics, relationship type, and corporation profile.

The detailed roles of agent based web services are as follows

R-Agent: Receives a customer's order and sells the product if it is in the inventory. If the order cannot be fulfilled by the retailer, the product is ordered from the manufacturer (M-Agent) and backlogged. The R-Agent pursues the goal of the retailer. One of the major decisions is how much to order from the M-Agent.

M-Agent: Manufactures products by assembling components. It receives product orders from the retailer (R-Agent) and places component orders to the supplier (S-Agent). The M-Agent pursues the goal of the manufacturer. The major decisions are how much to produce and how much to order from the supplier (S-Agent). The M-Agent must also decide whether to expand its supplier base if it faces supply uncertainties.

S-Agent: Produces components. It receives raw materials from the outside and supplies the components to the manufacturer (M-Agent). The S-Agent pursues the goal of the supplier. One of the major decisions is how much to produce.

MACE-SCM: Facilitates an additional level of collaboration and information sharing. The main priority of MACE-SCM is to maximize overall supply chain profits by assisting the other three agents and feeding them superior information gleaned from its knowledge base. MACE-SCM may or may not rely on the case base. The case base contains a decision set of competitors, environmental data such as total demand, economic rate, output data (market share), total revenue, and total cost. The coordinator MACE-SCM allows the R-agent, the M agent and the S-agent to access the case base.

5.4.2. COORDINATION SCHEME IN AGILE SUPPLY CHAIN BASED ON MULTI AGENT SYSTEMS (Ping Lou, 2005)

This section presents two different types of coordination mechanisms for two different types of agents based on Ping Lou's work.

The agile supply chain, composed of geographically dispersed manufacturers, supplier, distributors, is a distributed system, which can reconfigure and decompose with the dynamic alliance forming and decomposing. It is an operational strategy focused on inducing velocity and flexibility in the supply chain, thus enabling it to be customer centric. Agents can enter or leave the multi-agent system for the object, that is, they can be dynamically organized based on control and structure. The different functional agents in the multi-agent-based ASCM collaborate with each other for reconfiguring quickly and optimizing the supply chain management. With coordination among agents, the MAS achieve the goal of "the right products in the right quantities (at the right place) at the right moment at minimal cost". So coordination mechanism for ASCM should be able to represent and respond to real-time changes. The mechanism should be able to take uncertainty into account proactively and focus on producing robust coordination, that is, coordination that are resilient enough to absorb sources of uncertainty without being invalidated.

According to this paper, there are two types of agents, co operative and self-interested agents in a agile supply chain. The co operative agents attempt to maximize the profit of the supply chain as whole and are willing to incur any personal losses, while the self interested agents want to maximize their own profit, while not caring about the other members of the supply chain. For example, different production cells within an enterprise can act as cooperative agents. They attempt to minimize production costs and maximize the revenues of the enterprise as a whole, sometimes accepting local losses in order to facilitate production at other cells.

5.4.2.1. A Coordination Mechanism for Co operative Agents

Each node (enterprise) in a multi-agent-based ASC is represented by a multi-agent subsystem. The different functional agents in an enterprise cooperate with each other to achieve some specified producing tasks. The goal of cooperation among agents is to schedule optimally for material resources, manufacturing resources and human resources, so that this multi-agent subsystem becomes a collaborative system. In such a subsystem, every agent can lose its profit for global profit. Therefore, a coordination mechanism of decentralized scheduling and centralized decision-making (DSCDM) is used for it. The main ideal of this coordination mechanism is "as decentralized as possible, as centralized as necessary", the decentralized can achieve for responding quickly according to any change and the centralized can achieve for optimizing globally. In order to achieve the coordination mechanism, the contract net protocol combining with case-based reasoning is applied to it.

5.4.2.2. A coordination Mechanism for Self Interested Agents

Self Interested agents do not care for the global profit/cost in the supply chain and are only concerned about their own profit. Their principle goal is to maximize profit or minimize cost. There is no authority in a supply chain, so the DSCDM is improper for it. Therefore, the hierarchical distributed coordination mechanism (HDCM) with two phases, namely strategic-level coordination and operational coordination is proposed in this paper.

Strategic Level Coordination

Strategic level coordination is the re allocation of cooperative profits by contract. In a supply chain, all the members will keep two kinds of attitudes to fulfill their own activities, namely the cooperative and the individual. The cooperative means that the members collaborate with each other to form an alliance for satisfying the change of customers' requirements, while the individual means that each member fulfills his own activities all by himself. In the strategic-level coordination, there are two problems which need to be solved, one is the conditions of going to negotiation process for forming collaborated coalition; the

other is how to re-allot the profit so that all enterprises in the supply chain can build stable collaborated relationship.

Operational Level Coordination

In Operational Level Coordination the main goal is how to coordinate among the agents belonging to different members for completing a task which cannot be performed by a single agent. In order to achieve this goal, the following steps need to be taken

- Find an agent's coalition to finish the task optimally and
- How to solve the conflict between the personal interest of agents and the collective interest of agent's coalition.

The conflict can be solved by using the strategic level co ordination. So at the Optimal level one should find the coalition for an agent.

5.4.3. INTEGRATED MULTI AGENT BASED SUPPLY CHAIN MANAGEMENT (Frey, et al, 2003)

In a Supply Chain since necessary data are not available across the whole chain, an integrated approach for production planning and control taking into account all the partners involved is not feasible. This section explains how Multi Agent Systems can be used to solve this problem, based on the work done by Daniel Frey and others and presented in the paper "Integrated Multi-agent-Based Supply Chain Management" Successful integration of numerous MAS that perform both inter- and intra organizational planning and execution tasks is important for supply chain planning and execution.

5.4.3.1 Integrated Supply Chain Management Architecture

The different MAS and their functions, that are considered in this paper are, DISPOWEB (SCM scheduling), KRASH, IntaPS and FABMAS (shop floor Production planning and control), ATT/SCC (proactive tracking and tracing services). These MAS are integrated into a reference model comprising of a manufacturer of agricultural equipment.

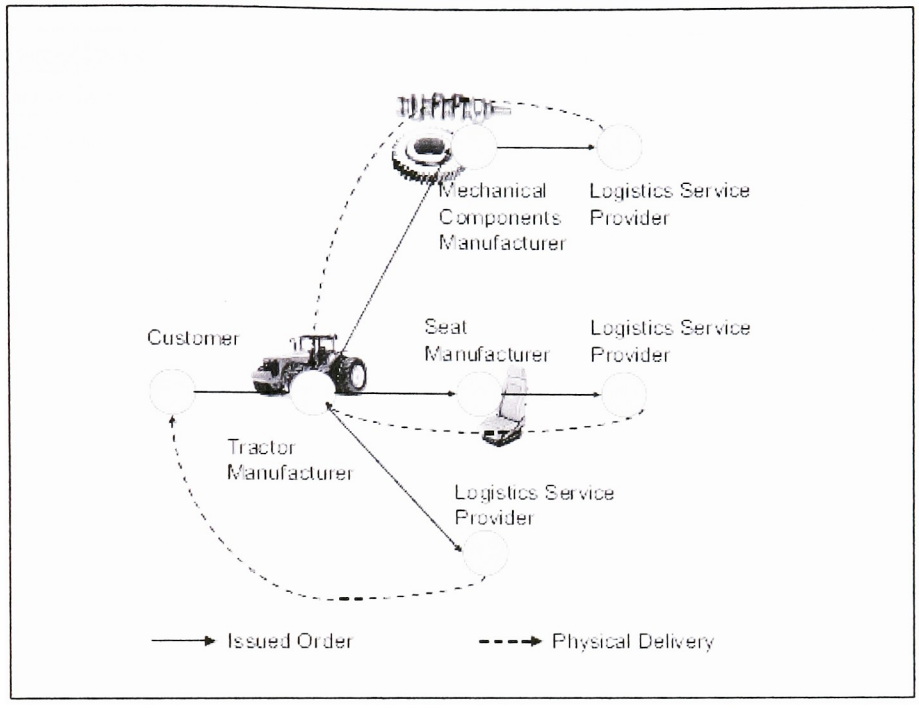


Figure 5.2: Supply Chain Scenario ***(Frey, et al, 2003)***

The manufacturer produces products by assembling parts that it procures from different suppliers. In this example a manufacturer of mechanical components and a supplier of seats is incorporated. The logistics service providers is not considered in the integrated SCM architecture.

Managing a supply chain requires many different tasks such as planning, execution and controlling of production, transportation and warehousing processes. These tasks require multiple MAS specializing on certain tasks to interact.

The following table gives the functionality of various MAS involved.

Table 5.1: Functionality of various MAS

Main Functionality	Project
Negotiations between enterprises	DISPOWEB
Integrated process planning and scheduling (with focus on discrete manufacturing)	IntaPS
Production planning and controlling (with focus on assembling industries)	KRASH
Production planning and controlling (with focus on batch production)	FABMAS
Operational tracking of orders including suborders in supply chains	ATT*
Analysis of historical tracking information (tracing)	SCC*

** ATT and SCC are conducted in one project*

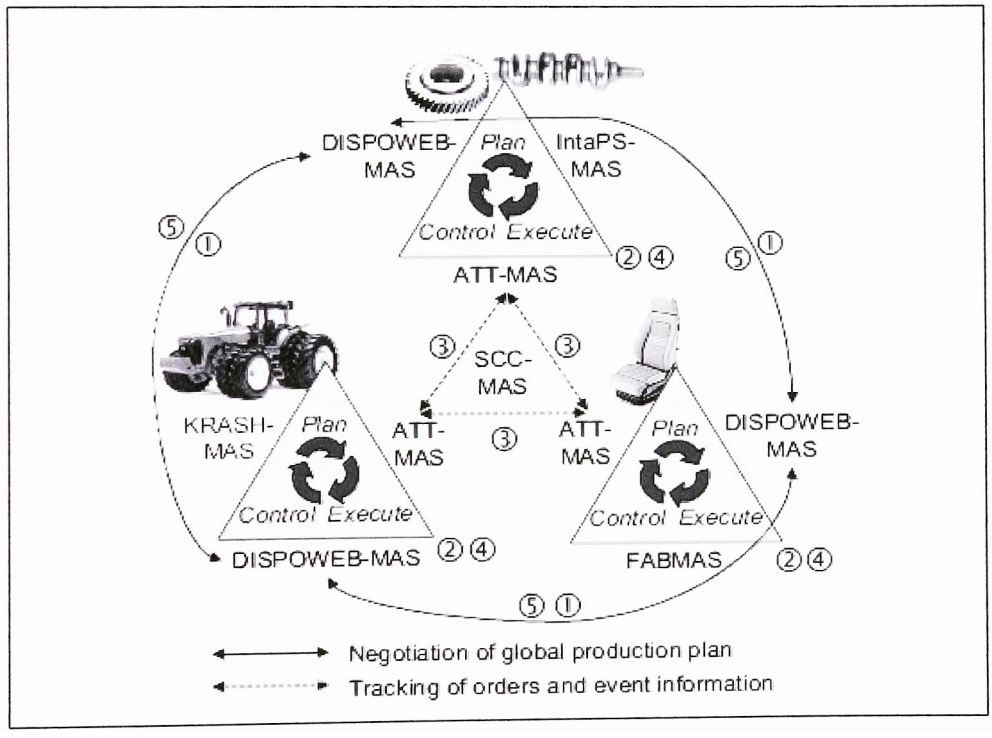


Figure 5.3: MAS Interaction in Integrated Supply Chain Management ***(Frey, et al, 2003)***

Distributed global planning is the first step in the planning process. DISPOWEB system is used to generate a initial plan of orders and suborders concerning prices and time-points of delivery. Once this is done software agents located at the different supply chain partners carry

on negotiations to optimize the costs and due dates of deliveries (1, Fig 3). These optimized delivery plans are used on intra-organizational level in each enterprise to plan the production of goods on each stage of the supply chain in detail. Three different multi-agent systems are concerned with varying aspects of production planning (2). They require the input from the DISPOWEB agents and generate detailed plans for their production facilities. These plans are the initial input for a controlling system, which is developed in the ATT/SCC project. This multi-agent system monitors orders on every stage of the supply chain using a distributed architecture to proactively detect events that endanger the planned fulfillment.. In case of an event (e.g. a disruption in a production line) the ATT system is engaged in a communication with the related partner enterprises and informs them of the event (3). This output can be used as a trigger for rescheduling plans on an enterprise level (4) or, in case of major events, even in the re-negotiation of the contracts on the inter-enterprise level of the DISPOWEB system (5). In addition to the operational monitoring of orders, the ATT system communicates results of negotiations to a trusted third party service called the SCC MAS. This agent system analyses the history of orders and their related sub-orders. SCC is able to identify patterns in the supply chain and order types that typically lead to problems during fulfillment. This information is both used as an input to enhance the tracking functionality of the ATT systems, as well as an input to the DISPOWEB agents that they can use for enhancing their negotiation strategies.
Therefore, DISPOWEB, KRASH and INTAPS are responsible for scheduling and SCC/ATT do tracking and supervision.

5.4.3.2. Interfaces and Gateways

The architecture explained above requires the interactions of the single components of the SCM reference model, where each MAS provides certain functionality. The exchange of data between the individual systems is done using standardized interfaces and gateways that arise from the corresponding functional specifications. The goal of the integrated SCM architecture is the coordination of the PPC (Production Planning and Control) activities throughout the whole logistics chain to achieve global monetary optimization. Consequently, the definition of the interfaces is based on a PPC database structure as shown

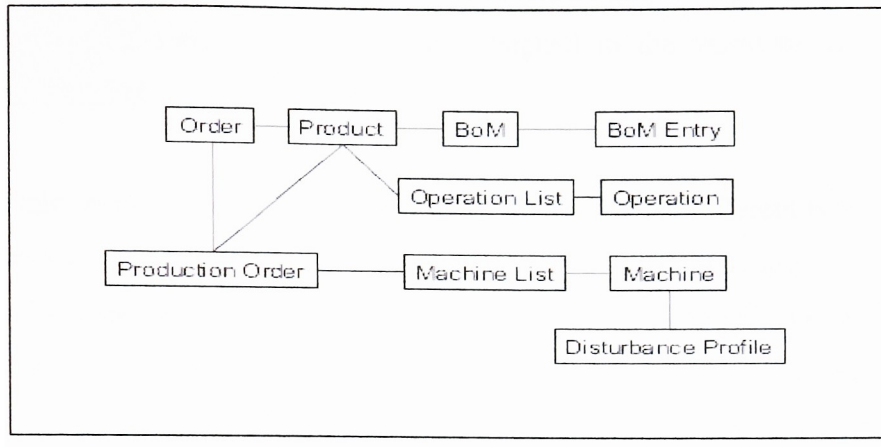


Figure 5.4: Database structure ***(Frey, et al, 2003)***

The table Product contains product-specific data. The bill of materials (BoM) lists all parts that are necessary to assemble the product, whereas the operation list represents the single work steps of the product assembly. Operation is a table that is used to assign operations to machines that are able to perform a particular work step (including potential setup processes). The Order table lists all customer orders. Besides the product and the quantity, the due dates or starting dates of the orders are defined here. Based on the orders and the other parameters, the Production Order table records the results of the planning process (production dates, facilities, quantities). The Disturbance Profile table is machine-specific and is based upon disturbance histories gathered from an MDA (Machine Data Acquisition) or a PDA (Production Data Acquisition) system and rules of thumb.

5.4.4. INTELLIGENT MULTI AGENTS FOR SUPPLY CHAIN MANAGEMENT (Yung and Yang, 1999)

This section explains the use of genetic algorithms to find the optimum solution to the problem of delivery distribution. It is based on the work done by Stanley K. Yung and Christopher C. Tang in their paper "Intelligent Multi-Agents for Supply Chain Management. In this paper the authors describe a supply chain model based on constraint network and propose a solution to optimizing the delivery schedule.

A constraint network is composed of a set of variables, a domain of potential values for each variable and a set of constraints. Values are assigned to the variables such that all the constraints are satisfied.

In a supply chain there are different operational models to handle different types of processes. The authors focus on the warehouse inventory system. In general, there are three types of Independent-Demand Inventory Systems, (1) Economic Order Quantity, (2) Continuous Review System (Q System) and (3) Periodic Review System (P System). Economic Order Quantity is the lot size that minimizes total annual inventory holding and order cost. The lot size is the order quantity from warehouse to upper stream of the supply chain. Continuous Review System (Q System) reviews the remaining quantity of an item each time a withdrawal is made from inventory, to determine whether it is time to reorder. Periodic Review System (P System) reviews the inventory position of an item periodically and a new order is placed at the end of each review. The number of periods between orders is fixed. For this discussion Q system of inventory management is selected.

The following is the constraint model for Q system:

IP = OH + SR - BO --- (1)
IP = inventory position of the item (in units)
OH =number of units in on-hand inventory
SR = scheduled receipts (open orders)
BO = number of units either back ordered or allocated

R = D + B --- (2)
R = reorder point
D = average demand during lead time L
B = safety stock
In the above model, when Ip is smaller than R, then reorder will be taken place. Thus, there is a constraint, IP > R. The constraint network is as shown.

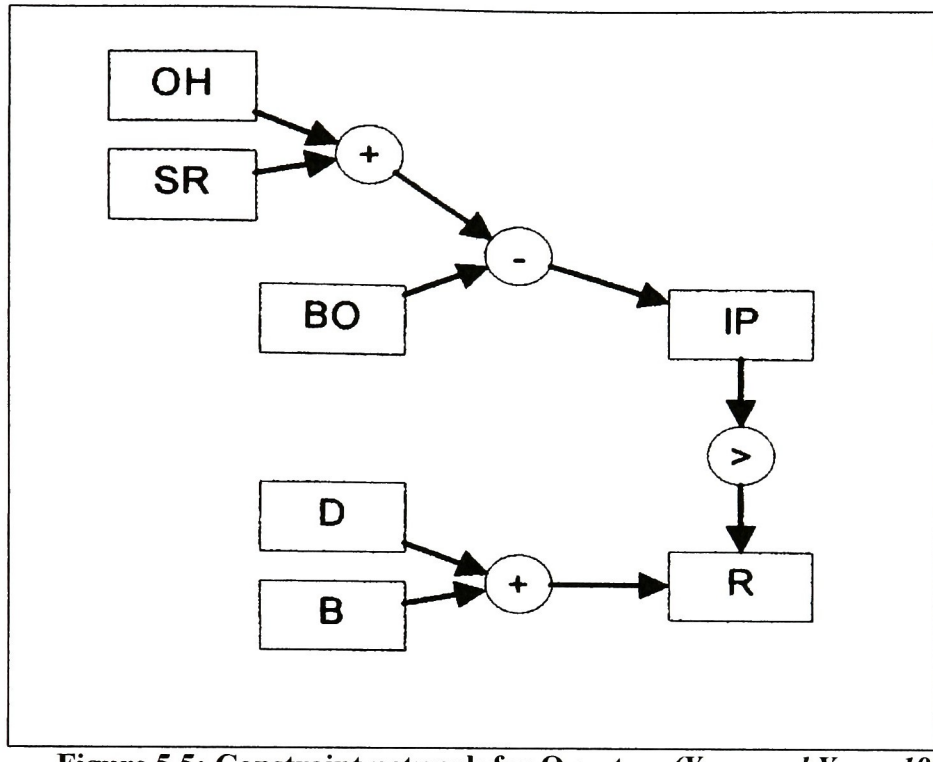


Figure 5.5: Constraint network for Q system ***(Yung and Yang, 1999)***

In a supply chain there is at least one warehouse at each location. Every warehouse has its own inventory management system to cope with its environment. Even if the same inventory system is applied there are differences in the parameters or constrain toughness. Constraints can also be applied to different areas in a supply chain, like resource allocation problem, material purchasing problem etc. When all supply chain components are modeled by constraints they can be linked together to form a global network and global optimization can be performed.

Consider the problem of how warehouses allocate products to customers. Products can be delivered to customers from one warehouse or from different warehouses at the same time. The task is to obtain a distribution pattern with minimum cost. Here the role of the agents is to gather information for optimization. The authors suggest two methods for agent communication and optimization, based on past research, which are negotiation method and mediated agent method. The mediated agent is selected to perform the genetic algorithm, for optimization.

5.4.5. GENETIC ALGORITHM (Yung and Yang, 1999)

This is a brief overview of how genetic algorithms can be used to do optimization without going into details.

To employ genetic algorithms, first represent the candidate solutions as chromosomes in a string of building blocks, genes. To ensure that the constraints are fulfilled, initialization will generate a population of chromosomes that can satisfy all the constraints. Then crossover and mutation generates a set of offspring. The offspring is checked with all the constraints after reproduction. A new population will be selected from the old population and the new offspring that depends on a predefined optimization function. Crossover, mutation and selection will be performed on the new population recursively until the population is converged, the resources are exhausted or the degree of optimization is satisfied.

Quantizing the delivery problem to two warehouses, warehouse 1 and 2 and four products A, B, C and D in a single order with quantities 2,5,8,10 respectively. The format of the chromosomes depends upon the number of warehouses, the number of products in a single order and the quantities of the product. Since a gene can only store 1 or 0; the number 2, 5, 8 and 10 are converted to binary numbers. Thus, the format of a chromosome will be as follow:

Format:

	Warehouse 1				Warehouse 2		
A	B	C	D	A	B	C	D
00	011	0001	01010	10	010	0111	00000

Figure 5.6: Chromosome Format

Crossover:

Portions of the building blocks (genes) of the selected parents (chromosomes) are exchanged to form new offspring. For the chromosome format above, the genes are exchanged or are not

changed at each product quantity randomly but exchanginggene positions are the same for each warehouse. The cutting point for exchange is also randomly selected for each product. The following figure will demonstrate a simple crossover with the above chromosome formats.

Parent 1	00	011	0001	01010	10	010	0111	00000
Parent 2	10	010	0111	00000	00	011	0001	01010
Offspring 1	00	010	0111	01010	10	011	0001	00000
Offspring 2	10	011	0101	00000	00	010	0111	01010

Figure 5.7: Example of Crossover

Mutation:

A random building block (gene) of a randomly selected parent (chromosome) is picked, and its value changed. For the chromosome format above, the gene is changed or is not changed at each product quantity randomly but exchanging gene positions are the same for each warehouse. The changing gene is also selected randomly for each product. The following demonstrate a simple mutation with the above chromosome format.

Parent 1	00	011	0001	01010	10	010	0111	00000
Offspring	00	010	0001	01000	10	011	0111	00010

Figure 5.8: Example of Mutation

Selection:

A weighted random selection method is used. The more optimized a member is in the group, more are its chances of being selected. When the population converge, the best solution is selected according to the optimization function.

5.5. EXAMPLES OF AGENTS IN INDUSTRY

This section presents some examples of agents used in different industries. It shows the type of agent, its percepts i.e. events which are used to trigger the agent, it goals and the actions it takes to achieve the goal. It also shows the environment in which the agent is used.

Table 5.2: Agent Examples

Agent Types	Percepts	Actions	Goals	Environment	Reference
Medical Diagnosis System	Symptoms, findings, patients answers	Questions, tests, treatments	Healthy patient, minimize cost	Patient, hospital	(Stuart, et al, 1995)
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite	(Stuart, et al, 1995)
Part-picking robot	Pixels of varying intensity	Pick up parts and sort them into bins	Place parts in correct bins	Conveyor belt with parts	(Stuart, et al, 1995)
Refinery controller	Temperature, pressure readings	Open, close valves, adjust temp.	Maximize purity, yield, safety	Refinery	(Stuart, et al, 1995)
Interactive English tutor	Typed words	Print exercises, suggestions, corrections	Maximize students score on tests	Set of students	(Stuart, et al, 1995)

Table 5.2 Agent Examples (Continued)

Agent Types	Percepts	Actions	Goals	Environment	Reference
Open Sesame	User actions	Learn work patterns and automatically perform tasks accordingly	Automate users mundane tasks	Macintosh, User	(http://www.hermans.org/agents/h32.htm , 2007)
Hoover	Type of information required by the user.	Sort the information according to requirement	Provide a single user interface to multiple information media.	Internet	(http://www.hermans.org/agents/h32.htm , 2007)
Internet Softbot	Queries from the user	Accesses only structured information (library database)	Provide the user requested information	Internet	(http://www.hermans.org/agents/h32.htm , 2007)

Table 5.2 Agent Examples (Continued)

Agent Types	Percepts	Actions	Goals	Environment	Reference
Advanced Plant Analysis and Control System	Nuclear plants feed water system	Data from plants main controller, feed water sensor and alarm values,	Maintain a steady plant state	Nuclear Plant	(Wang, 1997)
QinetiQ	User instructions, data from simulation of real world	Co ordinate the operation of multiple UAV's .	Let one operator control multiple platforms	UAV's	(Pechoucek, 2006)
The PowerMatcher	Power consumption, production cost	Control electricity production	Maximize profit by consuming electricity at low cost and generating it at high cost	Distributed Electricity Generation	(Pechoucek, 2006)
Adaptive, dynamic transport optimizer	Event management system inputs, tracking facility inputs,	Generate the optimal route using a real time route optimizer	Optimization and dispatching full/partial truckload, including tracking and real-time event handling		(Pechoucek, 2006)

Reference:

M. Wooldridge. "Agent Based Software Engineering". IEEE online proceeding, 1997.

Ghiyoung Im, Kun Chang Lee and Ohbyung Kwon. "MACE-SCM: An Effective Supply Chain Decision Making Approach based on Multi-agent and Case-based Reasoning". Proceedings of the 38th Hawaii International Conference on System Sciences, 2005.

Ping Lou, You-Ping Chen and Zu-De Zhou. "Study on Coordination in Multi-Agent-Based Agiles Supply Chain Management". Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005

Daniel Frey, Peer-Oliver Woelk, Roland Zimmermann and Tim Stockheim. "Integrated Multi-Agent-Based Supply Chain Management". Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03)

Christopher C. Yang and Stanley K. Yung. "Intelligent Multi-Agents for Supply Chain Management". IEEE 1999.

Hugos, M. Essentials of Supply Chain Management. John Wiley and Sons, Inc, 2003.

Russell Stuart and Peter Norvig. "Artificial Intelligence: A Modern Approach" 1995, Prentice-Hall, Inc.

"Examples of Agent Applications and entire agent systems" Retreived 2006 from http://www.hermans.org/agents/h32.htm

Huaiqing Wang, City University of Hong Kong, Chen Wang, University of Toronto. "Intelligent Agents in Nuclear Industry". IEEE Computer Society Press Los Alamitos, CA, USA. Volume 30, Issue 11 (November 1997) Pages: 28 – 34.

Michael Pechoucek, Czech Technical University. Simon G Thompson, BT. "Agents in industry". IEEE Computer Society, 2006

"Intelligent Agents" Retrieved 2006 from http://www.eil.utoronto.ca/profiles/rune/node6.html

6. MASCOT AGENT ARCHITECTURE

The agent architecture proposed in this thesis is based on MASCOT ("Multi – Agent Supply Chain Co ordination Tool") agent architecture. The information presented in this chapter is based on (Sadeh, et al, 1999). The chapter explains MASCOT architecture, how it is adapted using web service for this thesis and some of the advantages of using this architecture.

6.1. INTRODUCTION

MASCOT architecture is a reconfigurable, multilevel, agent based architecture for coordinated supply chain planning and scheduling designed to work in today's high-mix production environment and works to improve coordination across multiple facilities and to evaluate new product/subcomponent designs and business decisions. It is built around a customizable mixed- initiative agent wrapper. The wrapper provides a mechanism for communication amongst MASCOT agents operating at different levels of planning and scheduling and it also supports customizable mixed initiative planning.

6.2. OVERALL MASCOT ARCHITECTURE

The overall MASCOT architecture provides a framework for coordinated development and manipulation of planning and scheduling solutions at multiple levels of abstraction across the supply chain. Within this architecture, MASCOT agents serve as wrappers for planning and scheduling modules, each responsible for supporting the development and revision of planning, scheduling solutions for a particular group of facilities ata particular level of abstraction. Figure 6.1 presents an overview of the architecture illustrating the interaction between various types of multilevel MASCOT agents.

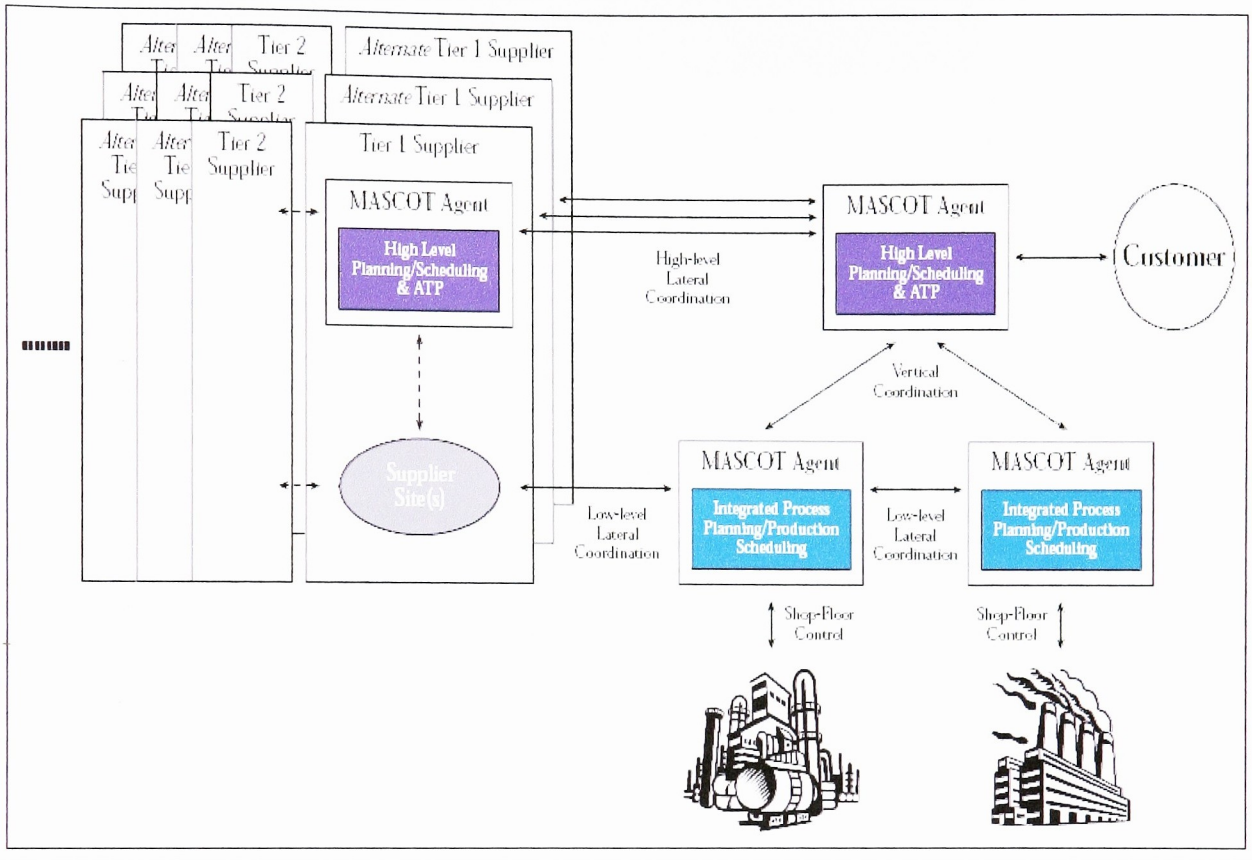


Figure 6.1: Overall MASCOT Architecture (Sadeh, et al, 1999) (Sadeh et al, 1999)

Lower level agents are wrappers for planning and scheduling modules that support single facilities over short to medium term horizons. Higher level agents are coordination wrappers for tactical or strategic planning and scheduling modules that generally require looking over longer horizons and across multiple facilities.

6.3. MASCOT AGENT ARCHITECTURE

MASCOT agent architecture is a blackboard architecture. Backboard architectures emphasize the modular encapsulation of problem solving knowledge within independent knowledge sources (KSs) that work collectively to develop solution to problems by communicating through the blackboard which is a shared data structure. Figure 6.2 presents the MASCOT wrapper agent architecture

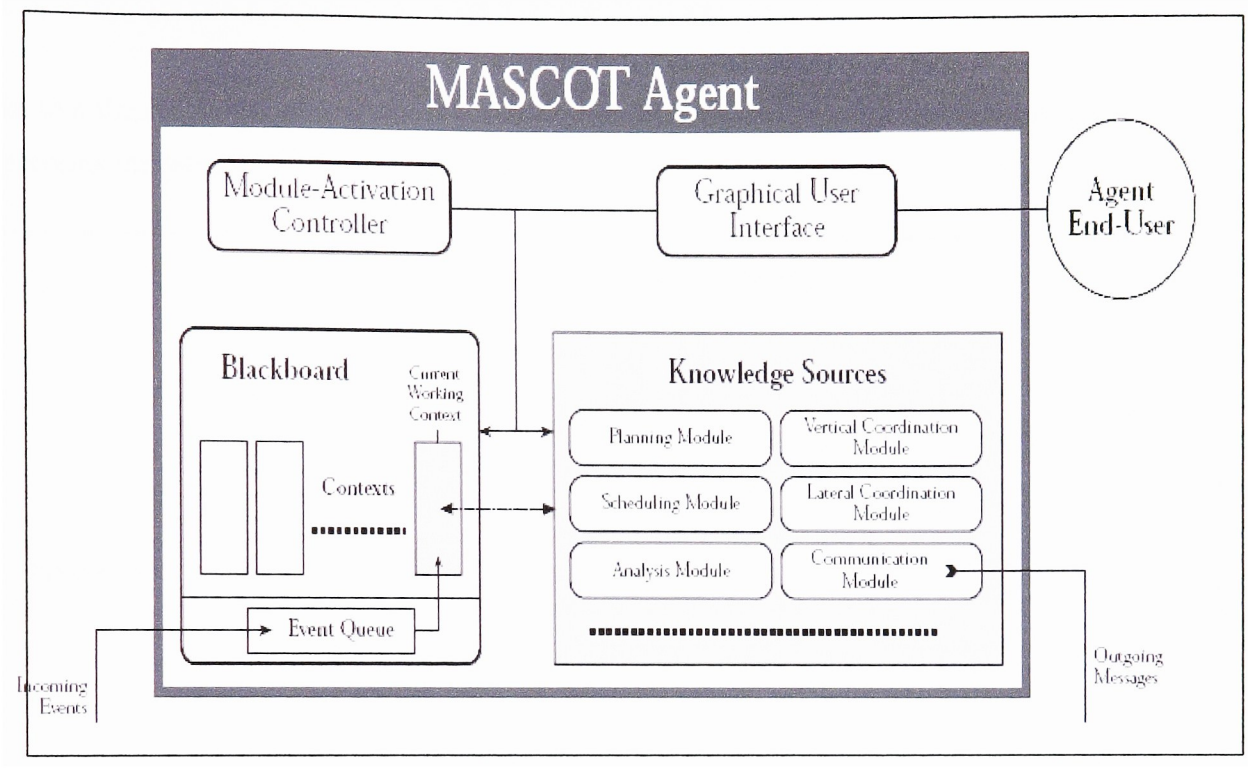


Figure 6.2: MASCOT wrapper agent architecture (Sadeh, et al, 1999) (Sadeh et al, 1999)

Each MASCOT agent includes:

- A set of planning, scheduling, analysis, coordination, and communication modules (knowledge sources). These modules are used appropriately to generate solutions to different problems.
- A blackboard, which serves as the repository of partial and complete integrated planning and scheduling solutions organized in different contexts. Any issues which need to be worked on are stored as a context on the blackboard.
- A module-activation controller that orchestrates the construction and revision of solutions through the activation of services supported by knowledge source modules, either based on direct input from the end-user or the agent's control heuristics or a combination of both.
- A graphical user interface.

6.4. PROPOSED AGENT ARCHITECTURE

In this thesis, we propose to use the MASCOT architecture as a reference frame. Figure 6.3 presents the proposed agent architecture.

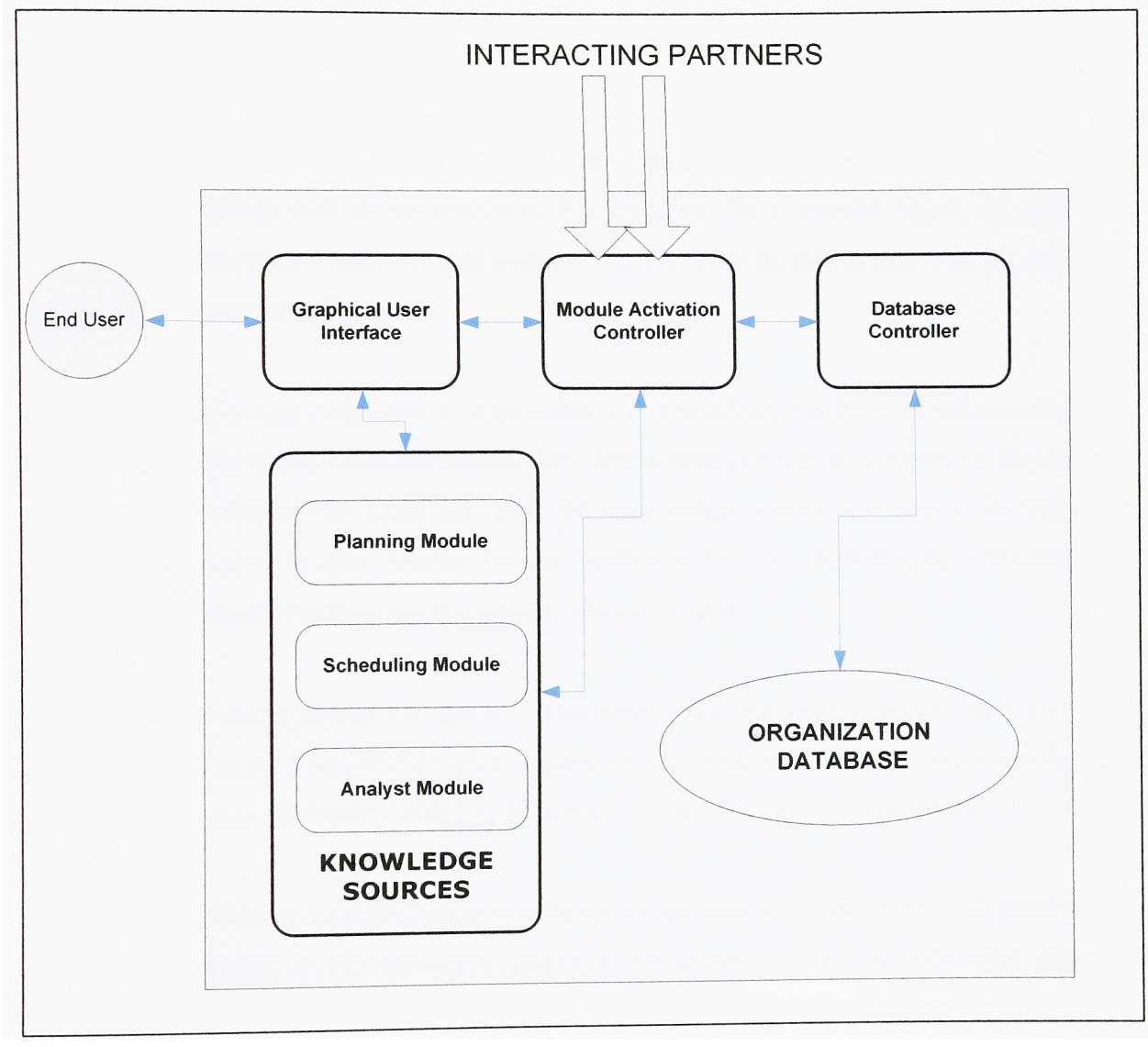


Figure 6.3: Proposed Agent Architecture

The main components of the proposed agent are:

- Graphical User Interface
- Module Activation Controller Module
- Database Controller Module
- Knowledge Source Module
- Organization Database

The graphical user interface (GUI) provides user interaction. It is the agent's mechanism to interact and simulate with its environment. For example, for a supplier agent, its graphical user interface provides a mean to demonstrate simulation of its interaction with its partners, namely manufacturers and carriers.

The Module activation controller is responsible for controlling the flow of information into and internal to the agent. Communication from interacting partners is received by the module activation controller of the agent and then the appropriate course of action to be taken, to generate a response, is also decided by the module activation controller by activating the correct solution modules from the knowledge source module.

Database controller controls the agent's interaction with the organizations database. Any information from the databased which is required to generate a response; is secured using the database controller. No other module of the agent has direct access to the database.

The knowledge source module is used to perform all the necessary calculations to generate an appropriate response to the interacting partners. Different calculations, like what price to offer, the earliest possible delivery date etc are performed in the knowledge source module.
The organization database is used to store all the information of the organization.

Web service technology is used to implement the module activation controller, database controller and the knowledge source modules of the agent.

Reference:

Norman M. Sadeh, David W. Hildum, Dag Kjenstad,and Allen Tseng. "MASCOT: An Agent-Based Architecture for Coordinated Mixed-Initiative Supply Chain Planning and Scheduling". Third International Conference on Autonomous Agents (Agents '99) Workshop on Agent-Based Decision Support for Managing the Internet-Enabled Supply Chain, Seattle WA, 1 May 1999.

7. PROPOSED ARCHITECTURE

The architecture proposed in this thesis demonstrates the use of agent technology in supply chain management. The main systems implemented are a manufacturer, a supplier and a carrier. A retailer also interacts with the manufacturer. The implementation adopts MASCOT agent architecture and demonstrates decision making in a supply chain. It shows how complex and tedious decisions in a supply chain can be automated using information technology. Open Applications Group Integration Specification (OAGIS) based standard business object documents (BODs) are used to implement communication amongst agents. OAGIS version 9.0 is an XML based standard proposed by Open Applications Group Inc, (OAGi) consortium. XML is an open standard, which can be used by anyone to suit their own requirements. Since OAGIS based BODs are XML based they are very convenient in terms of ease of use and availability. The different BODs can be used to construct different business scenarios which are present in industry. This chapter briefly explains the overall interaction amongst business partners, the internal structure of the agents and the decisions made by them.

7.1. BIG PICTURE

In a supply chain, inventory flows in the form of raw materials from the supplier to manufacturer and in the form of finished goods from manufacturer to retailer. A carrier is responsible for delivering the inventory between the two partners. A retailer places his order with the manufacturer for a product. Based on various factors, the manufacturer can either fulfill the order or generate a production schedule or place an order for the raw materials required from his suppliers. Thus, one can see that the order fulfillment process involves making many complex and tedious decisions and communicating it to the various partners.

In this thesis, each individual partner namely a supplier, manufacturer and carrier is represented as a complete independent agent, interacting with each other. This interaction is depicted in Figure 7.1. An example of such an interaction is the retailer enquiring about a specific product with the manufacturer by sending a "GetRFQ" message, the manufacturer

responding to this request by sending a "ShowRFQ" message. For the manufacturer to be able to generate the "ShowRFQ" message, he needs to take various decisions like what price to quote for requested product, checking if the desired delivery date can be meet etc. Complex decisions need to be taken by each agent for execution of any process like generation of "ShowRFQ" message. These complex decisions can be tedious, error prone and time consuming. Automation of these decisions can save an organization valuable time and prevent errors, factors which are very important for success in today's competitive environment.

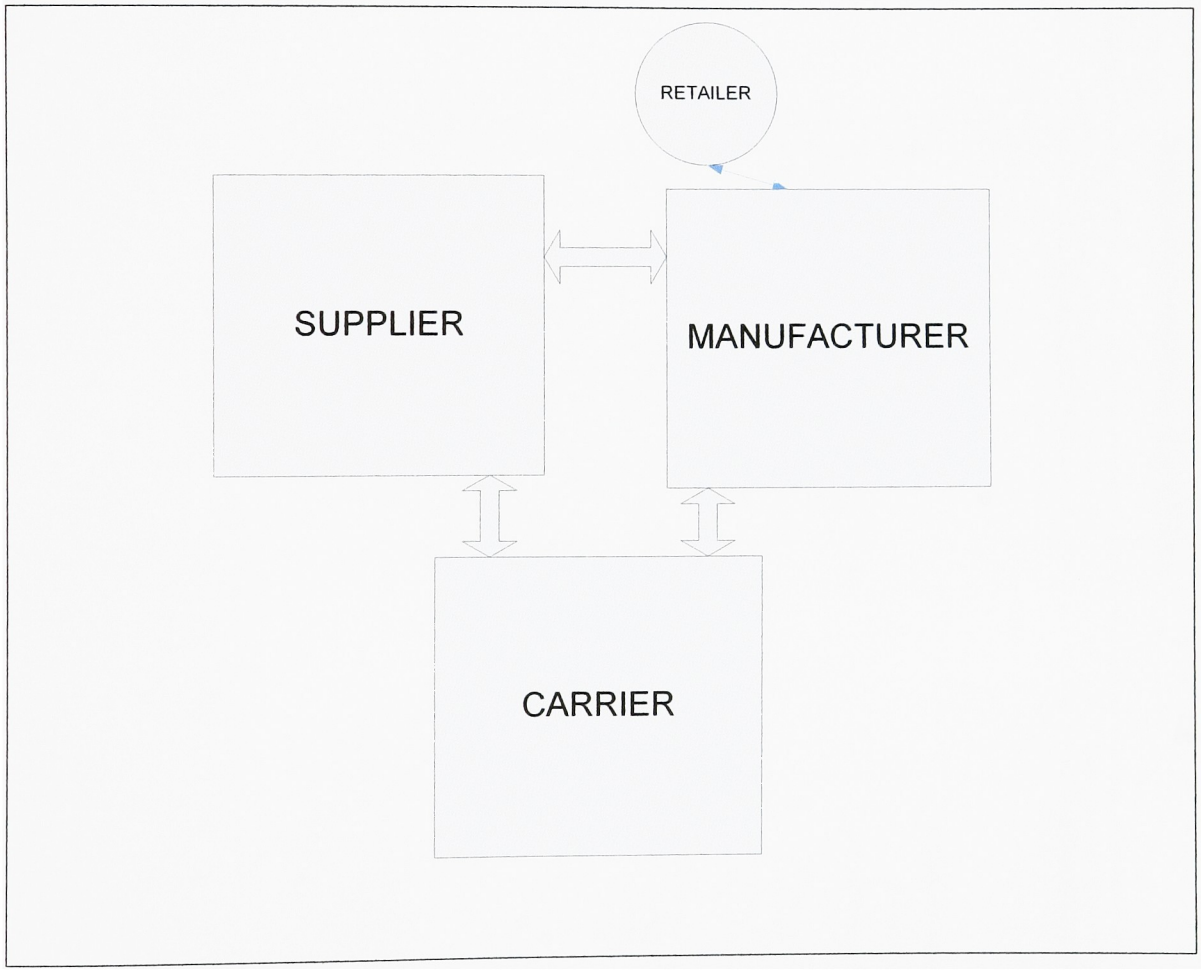


Figure 7.1: Thesis

All the possible interaction scenarios amongst each agent, implemented in this thesis, are shown in Figure 7.2.

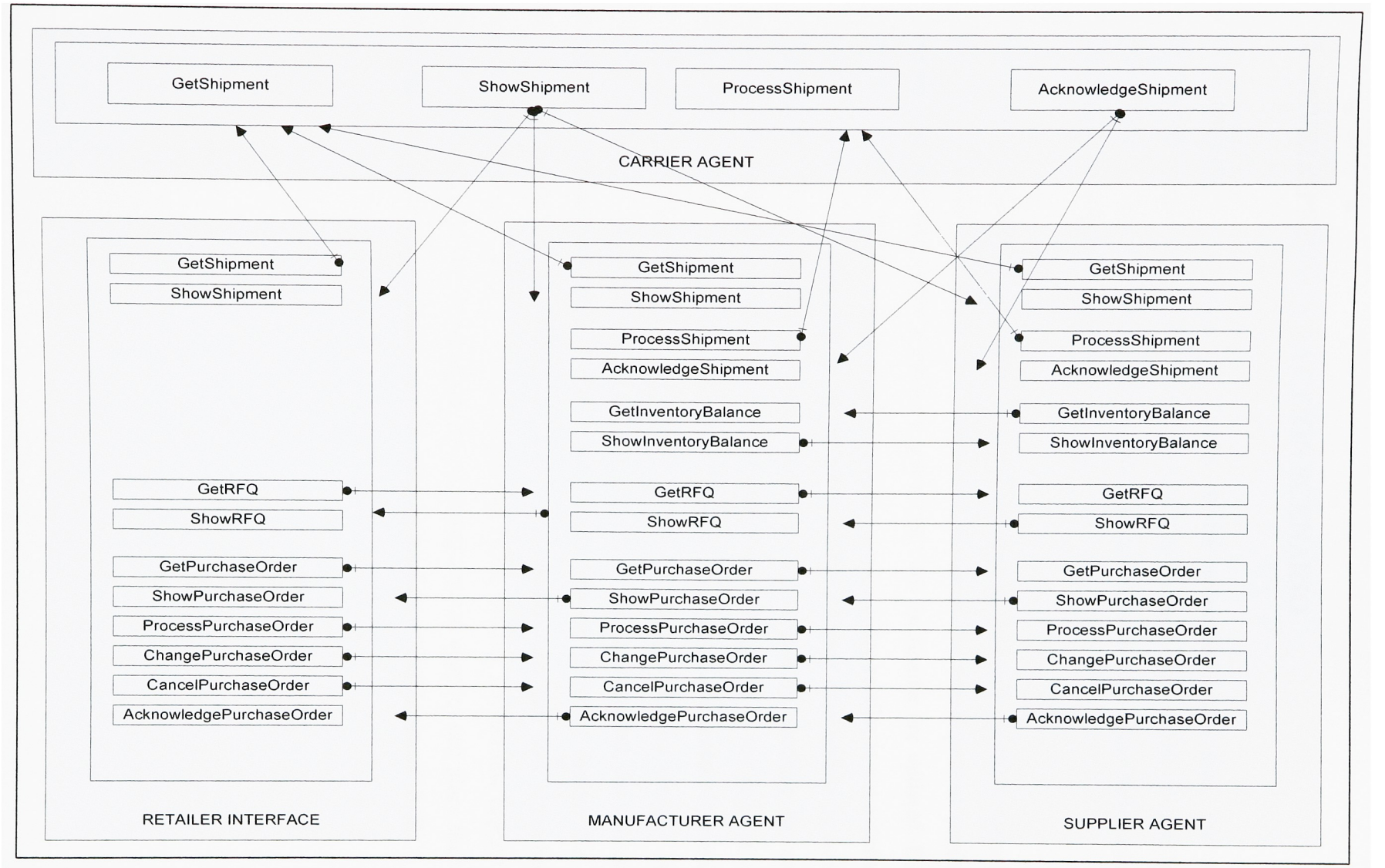


Figure7.2: Possible Interactions

7.2. DEVELOPED INTERACTION SCENARIO

Before proceeding to explain the details of the proposed architecture, the scenario implemented is briefly explained in this section. The detailed explanation of each business object document transferred, information exchanged and the decisions made during each stage is explained in detail in the preceding sections.

Interactions are initiated by a request for quote submitted by the retailer with the manufacturer agent. The manufacturer agent then checks his inventory for the requested product and depending on available quantity further negotiations with the retailer are initiated. Based on different inventory levels, a price is offered to the retailer. If the retailer is not satisfied with the offer, he can request a change in the price or the delivery date option by sending a "ChangeRFQ" BOD. The manufacturer responds to this request by checking the retailer's ratings. If the retailer does not have a poor rating then a reduction in the price is offered. When the retailer is satisfied with the RFQ, he can place his order by sending a "ProcessPurchaseOrder" BOD. The manufacturer acknowledges this by sending back an "AcknowledgePurchaseOrder" BOD. For the delivery of the consignment, the manufacturer sends a "GetShipment" BOD to the carrier. Depending on different factors the carrier responds by sending a "ShowShipment" BOD. The manufacturer can now place his order with the retailer by sending a "ProcessShipment" BOD. Similar interactions happen at the supplier agent also. At the carrier agent, the manufacturer and supplier can place their order. Also, they can inquire about their shipment status by sending a "GetShipment" BOD.

7.2.1. MANUFACTURER SYSTEM

The manufacturer system, represented by the manufacturer agent, interacts with retailers, suppliers and carriers. A retailer places his orders with the manufacturer. The manufacturer receives raw material from suppliers and carriers are used to deliver an order to retailers and also to receive raw materials from the supplier's. In this thesis, the internal architecture of the manufacturer agent consists of five different modules as seen in Figure 3.

- Manufacturer Graphical User Interface (MGUI).
- Manufacturer Module Activation Controller (MMAC).
- Manufacturer Database Controller (MDC).
- Manufacturer Knowledge Source (MKS) and
- Manufacturer Database

7.2.1.1. Manufacturer Agent Modules

- **Manufacturer Graphical User Interface (MGUI):**

  MGUI provides users with a mechanism to interact with the manufacturer agent. It is used to simulate the flow of information with the agent.

  Using MGUI, user's can simulate different scenario of retailer and supplier interaction with the manufacturer. This can be seen in the first two tabs ("Retailer" and "Supplier") in figure 7.4. For example, the user can simulate a retailer placing a "GetRFQ" business object document (BOD) with the manufacturer. He can input different values on the GUI. To this the manufacturer responds by sending a "ShowRFQ" BOD. The results of "ShowRFQ" BOD are displayed on the GUI, just as if they are viewed by the retailer system.

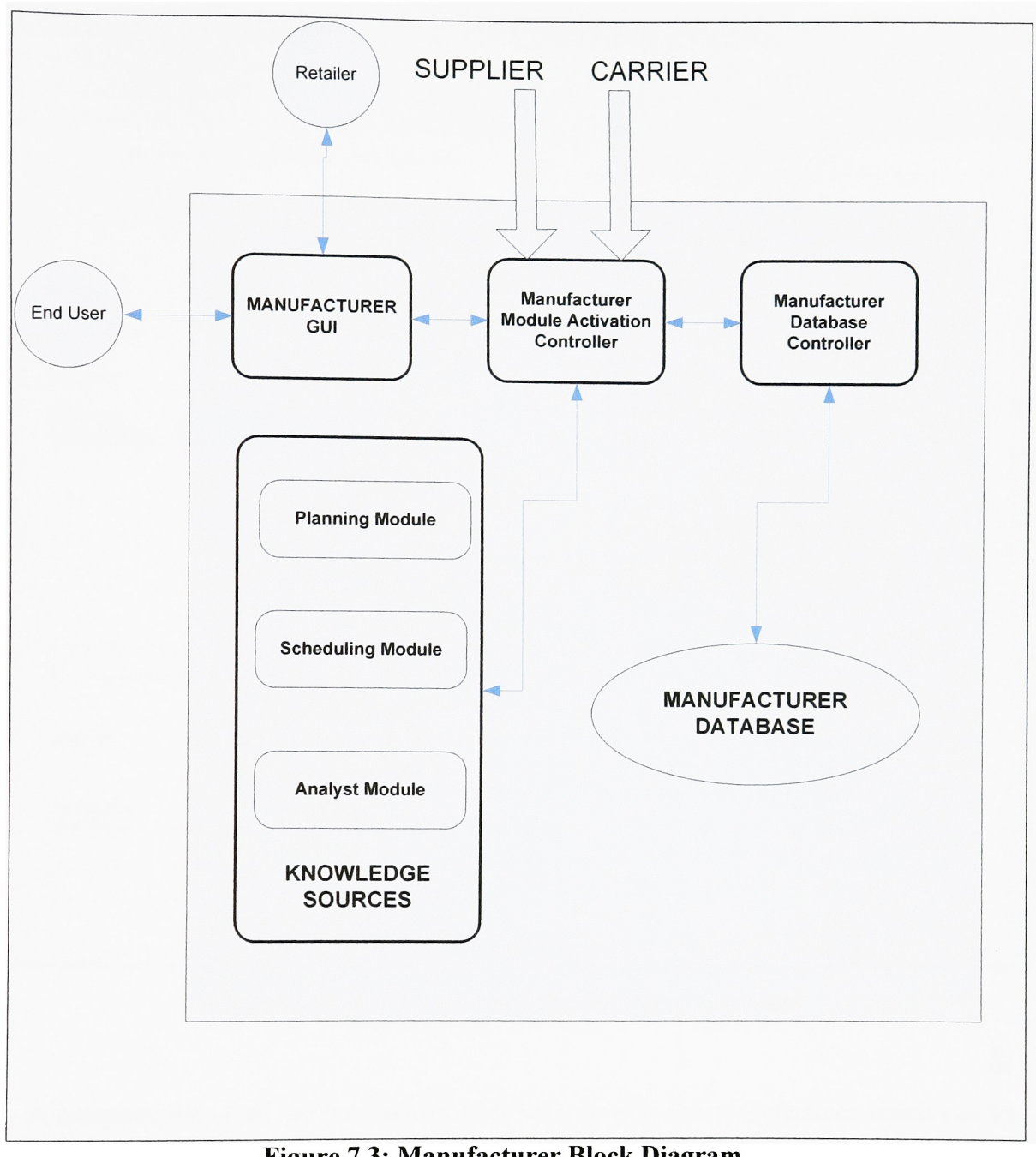


Figure 7.3: Manufacturer Block Diagram

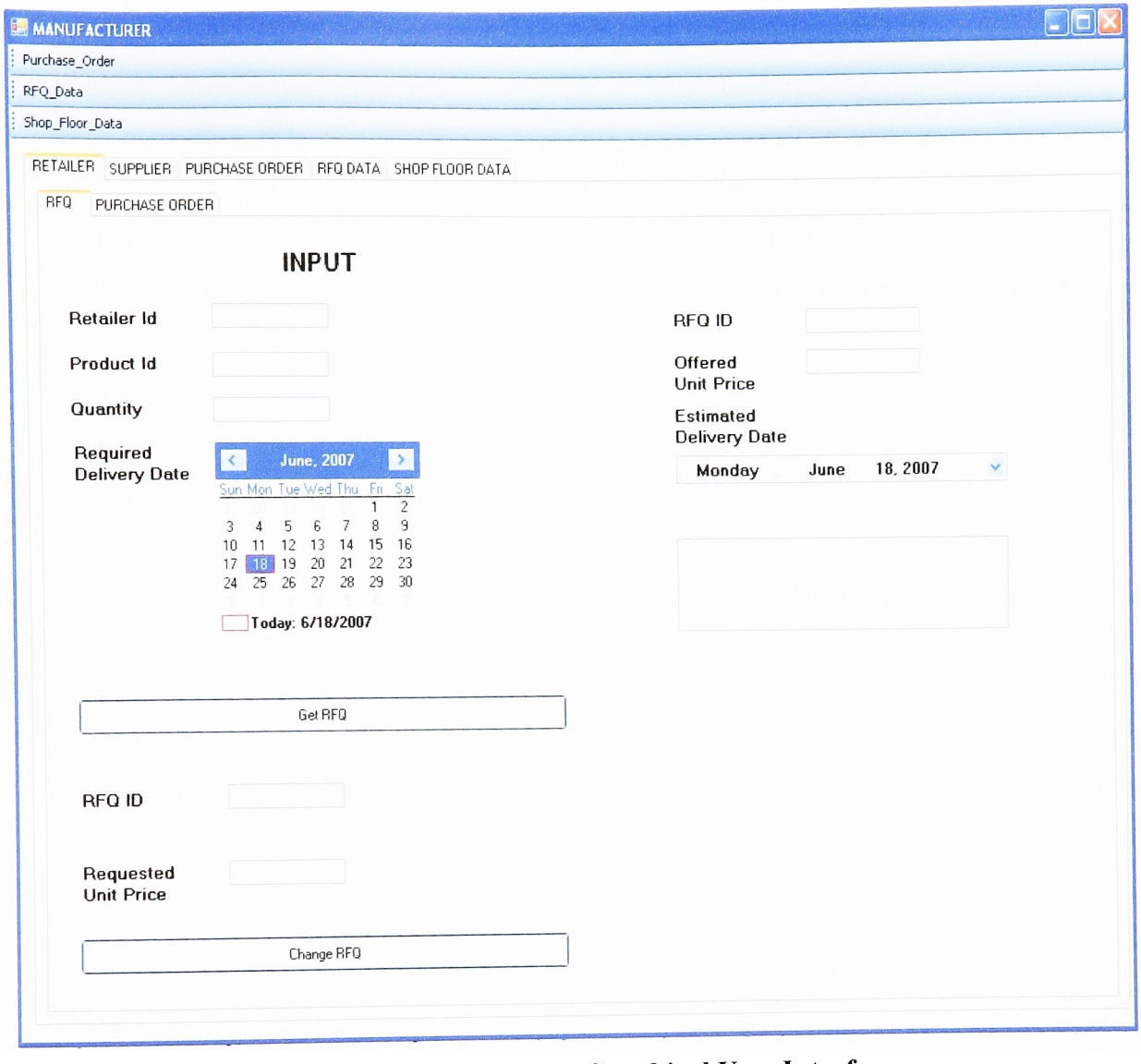


Figure 7.4: Manufacturer Graphical User Interface

A complete list of all the interactions between a retailer and manufacturer which can be simulated using this implementation are as follows

GetPurchaseOrder ↔ ShowPurchaseOrder,

ProcessPurchaseOrder ↔ AcknowledgePurchaseOrder,

ChangePurchaseOrder ↔ AcknowledgePurchaseOrder and

CancelPurchaseOrder ↔ AcknowledgePurchaseOrder interaction.

Supplier interacts with the manufacturer by sending "GetInventoryBalance" BOD for his product. Manufacturer responds by sending a "ShowInventoryBalance" BOD. In this implementation, it is not possible to stimulate carrier interaction with the manufacturer using MGUI, as the carrier does not initiate any interaction with the manufacturer. PURCHASEORDER, RFQ DATA and SHOP FLOOR DATA tabs are used to display relevant information from the manufacturer database. The logic used to prepare the values of the BOD which the manufacturer sends in response to a request from retailers and suppliers is explained later.

- **Manufacturer Module Activation Controller (MMAC)**

Manufacturer Module Activation Controller or MMAC is the main controller of the agent. It acts like a traffic controller and is responsible for taking appropriate actions on the requests from MGUI. It is responsible for receiving requests from retailers and suppliers and deciding what action to take for a response. For example, if a retailer places a GetRFQ BOD request through MGUI, the MMAC receives the request and retrieves all the information that is needed by the Knowledge Source Module (which is used for decision making) using the Database Controller Module (used for interaction with the database). After the Knowledge Source Module completes the decision making process, MMAC prepares ShowRFQ BOD and sends it back as a response to the retailer.

- **Manufacturer Database Controller (MDC)**

Manufacturer database controller is used for interaction with the database of the manufacturer. It is through the MDC that all the requests for information required for decision making are retrieved from the manufacturer database. Also, any updates required to be made to the data in the database are done using the MDC. The process of updating and retrieval of information is performed by MDC using SQL queries. For example, to decide the price to be offered for any product request through "GetRFQ", MMAC needs to know the unit price of that product. It places a request with the MDC which then retrieves the unit price from the database table "ManufacturerPriceData" and sends it back

to MMAC. This information along with other relevant information is then sent to the Knowledge Source module to decide the price to be offered. The MDC is the only module in the manufacturer agent which is allowed to interact with the manufacturer database. No other module in the agent can interact with the database.

- **Manufacturer Knowledge Source (MKS)**

Manufacturer Knowledge Source module is responsible for the process of decision making. Decisions like "What price to offer?", "Can the requested delivery date be meet?" etc. are taken in this module. MAC places a request with MKS to perform a decision. It also provides all the necessary information which is required by the MKS to take the decision. A detailed description of all the decisions performed by MKS is presented later.

- **Manufacturer Database**

The database is a place where all the data is stored. Manufacturer stores all his data in his database. In this thesis, SQL server 2005 express is used to implement the database.

7.2.2 INTERACTION DETAILS FOR MANUFACTURER AGENT

Retailers and suppliers interact with the manufacturer agent. During these interactions which messages are transferred, what information is exchanged, different decisions taken are presented in the tables 7.1 – 7.7.

7.2.2.1. Request for Quote

A retailer would like to know what price the manufacturer is offering a particular product for. For this purpose a retailer uses a "GetRFQ" BOD to request to the manufacturer, the quote for a particular item.

Table 7.1: GetRFQ interaction between manufacturer and retailer

No.	Retailer	Manufacturer	Decisions taken
1.1	GetRFQ The retailer sends this message to place a request for quote for a particular product. Retailer also sends the quantity required and the delivery date on which it would like to receive the delivery if it places an order for it.		
1.2		ShowRFQ MAC receives GetRFQ from retailer. MKS then decides if it should continue interaction with the retailer. If interaction is to be continued then MKS decides the price to quote for the requested product and quantity. MKS also checks if desired delivery date can be meet using the current inventory, if not then can the product be manufactured by the requested delivery date. If it cannot be manufactured by the requested delivery date then what is the earliest possible delivery date by which the requested order	➢ Should we continue negotiation with the retailer? ➢ Is there is enough inventory to complete the requested order? ➢ What is the earliest possible delivery date by which we can meet the order? ➢ What price to offer?

		can be meet. If negotiation is not to be continued then the retailer is informed accordingly. MAC sends this information to the retailer using a ShowRFQ BOD. This information is recorded in its database.	

The retailer would like to request a change in the price offered by the manufacturer. For this the retailer uses the "ChangeRFQ" message.

Table 7.2: ChangeRFQ interaction between manufacturer and retailer

No.	Retailer	Manufacturer	Decisions taken
2.1	ChangeRFQ The retailer uses the RFQ number it received during GetRFQ transaction from the manufacturer to place a request to change the price offered by the manufacturer for a product by sending the ChangeRFQ BOD		
2.2		AcknowledgeRFQ The manufacturer receives the ChangeRFQ request from retailer. He then checks the status of the retailer and based on the status information it offers a new price to the retailer using the AcknowledgeRFQ	➢ What new price to offer?

		BOD. The manufacturer also checks if a price reduction has been offered before. If yes, then no further reduction in price is offered. It records this information in his database.	

7.2.2.2. Purchase Order

When a retailer wants to place an order for goods from the manufacturer, he uses a "GetPurchaseOrder" BOD. In this application, the retailer needs to have an RFQ number as a reference to proceed with purchase order transactions.

Table 7.3: GetPurchaseOrder interaction between manufacturer and retailer

No	Retailer	Manufacturer	Decisions taken
3.1.	GetPurchaseOrder The retailer initiates the process of placing an order by sending this message to the manufacturer. Retailer sends the RFQ number it received for the product during RFQ transaction		
3.2		ShowPurchaseOrder The manufacturer retrieves the RFQ number from the GetRFQ message and retrieves all the relevant information from its database related to the RFQ transaction and sends this information back to the retailer	

		using ShowPurchaseOrder message.	

Now, the retailer can go ahead and place the order using a "ProcessPurchaseOrder" message. However, if the retailer wants to change the purchase order he can use the "ChangePurchaseOrder" message. The retailer cannot change a purchase order once it has been processed and agreed upon using the "ProcessPurchaseOrder" message.

Table 7.4: ChangePurchaseOrder interaction between manufacturer and retailer

No	Retailer	Manufacturer	Decisions taken
4.1	ChangePurchaseOrder If the retailer wants to change the quantity desired or the delivery date required, then he can use this message.		
4.2		AcknowledgePurchaseOrder The manufacturer then decides if it can allow the change requested by the retailer and conveys the result to the retailer. It records the changes in its database.	➢ Allow the change? ➢ Is there is enough inventory to complete the requested order? ➢ What is the earliest possible delivery date by which we can meet the order?

Table 7.5: ProcessPurchaseOrder interaction between manufacturer and retailer

No	Retailer	Manufacturer	Decisions taken
5.1	ProcessPurchaseOrder The retailer uses this message to place an order. Purchase order number is also sent as a reference.		
5.2		AcknowledgePurchaseOrder The manufacturer accepts the request to process the order and records this information in its database.	

Once the order is placed, if the retailer decides to cancel the order then it can try to do so using the "CancelPurchaseOrder" message.

Table 7.6: CancelPurchaseOrder interaction between manufacturer and retailer

No	Retailer	Manufacturer	Decisions taken
6.1	CancelPurchaseOrder Retailer uses this message to try and cancel the purchase order Purchase order number is sent as reference		
6.2		AcknowledgePurchaseOrder The MAC receives this request and using the MKS decides what penalty to charge the retailer for canceling the order.	➢ What should be the penalty?

7.2.2.3. Inventory Balance

Using the MGUI users can also simulate the interaction of suppliers with the manufacturer. In this implementation, the supplier initiates a "GetInventoryBalance" message with the manufacturer so that it can keep a check on the current stock levels of the manufacturer and plan his production accordingly, by anticipating the next order from the manufacturer.

Table 7.7: GetInventoryBalance interaction between manufacturer and supplier

No	Supplier	Manufacturer	Decisions taken
7.1	GetInventoryBalance: A supplier uses this message to know the current inventory levels of the manufacturer. It sends the product number to the manufacturer.		
7.2		ShowInventoryBalance: The MAC retrieves the product number from the message and using MDC retrieves the current inventory level of the product. This information is then sent back to the supplier using ShowInventoryBalance BOD.	

7.2.3. DECISIONS TAKEN IN MANUFACTURER AGENT:

The decision making ability in manufacturer agent is present in the manufacturer knowledge source element. In this thesis implementation, the manufacturer knowledge source element can perform the following decisions

- Should we continue negotiation with the retailer?
- Decide the unit price to be offered?
- Decide the earliest possible delivery date?
- New updated price to be offered?
- Calculate the penalty for cancellation of a purchase order?

Should we continue negotiation with the retailer?

When a retailer places a "GetRFQ" message, the MMAC receives it and asks the MKS to decide if the manufacturer should continue negotiation with the retailer. If the status is "Poor" then further negotiation is refused else further negotiation is initiated.

Decide the unit price to be offered?

This decision is taken when a retailer places a request for quote for a particular item. The input factors that are considered while making this decision are the current inventory level of the product, the required quantity, the quantity ranges and the corresponding unit price which the manufacturer has set beforehand and a boolean value which indicates if the required quantity can be manufactured. First the required quantity is compared against the quantity ranges and the corresponding price is set as the unit price. Then a check of the required quantity is done against the current inventory level of the product and if the required quantity is less than the current inventory level then the set unit price is returned as result. However, if the required quantity is greater than the current inventory level and requires production of the product, then the unit price is increased by a amount equal to 15% of the previous unit price.

Decide the earliest possible delivery date:

Whenever the requested delivery date cannot be meet, this decision is required to be made to calculate the earliest possible delivery date when the order can be meet. The inputs to this

method are the current production, the capacity of the shop floor to manufacture the product in a production day, the requested quantity and the number of days the carrier would take to make the delivery. The number of days the carrier would take to make the delivery is an arbitrary number and can be changed. It is assumed, for simplicity, that only a single machine is available to make a particular product. The formula used is as follows

Possible Return Date = `((lineduprod/sum) + (quantity/sum) + carrierdays);`

Where `lineduproduction - current inline production,`

`sum =` capacity of the shop floor to manufacture the product in a production day,

quantity = requested quantity and

carrierdays = number of days the carrier would take to make the delivery.

New updated price to be offered:

This decision is taken in response to a request from a retailer to reduce the offered price. The offered price is shown to the retailer through the "ShowRFQ" BOD. If the retailer is not satisfied with the price offered to him, he/she can request a reduction in the price. There are two input factors to this method, the status of the retailer and the offered old price. The status of the retailer is based on his past record with the manufacture. There are four levels to the status namely:

New – if the retailer has no past history with the manufacturer and is interacting with the manufacturer for the first time.

Good – Retailer has a good record and has not cancelled more than 3 purchase orders

Very Good – If the retailer has not cancelled any purchase orders he has placed with the manufacturer.

Poor – If the retailer has cancelled more than 3 purchase orders

Based on these status levels, a decision to update the price is taken. If the retailer status is very good, then a 10% reduction in the old price is offered. If the retailer status is "Good" or "New" then a 5% reduction is offered. If the status is poor then no reduction is offered.

Calculate the penalty for cancellation of a purchase order:

If a retailer wishes to cancel a purchase order he/she has placed then this decision is made. The inputs to this decision are the promised delivery date and the offered unit price. The penalty calculation is based on the difference between the promised delivery date and the current date. If the difference is less than seven days then the retailer is not allowed to cancel his/her purchase order. If the difference is between seven to fourteen days then the retailer is charged a penalty of 30% of the unit price. And if the difference between the promised delivery date and the current date is more than fourteen days then the retailer can cancel his/her purchase order without incurring any penalty.

7.2.4. SUPPLIER SYSTEM

The supplier supplies raw material to manufacturers. The manufacturers initiate interaction with the supplier to place a request for a product. In this implementation, the supplier system is also based on MASCOT agent architecture. It comprises of the following modules:

- Supplier Graphical User Interface (SGUI)
- Supplier Module Activation Controller (SMAC)
- Supplier Database Controller (SDC)
- Supplier Knowledge Source (SKS) and
- Supplier Database

Block Diagram of supplier Module is as shown below in figure 7.5.

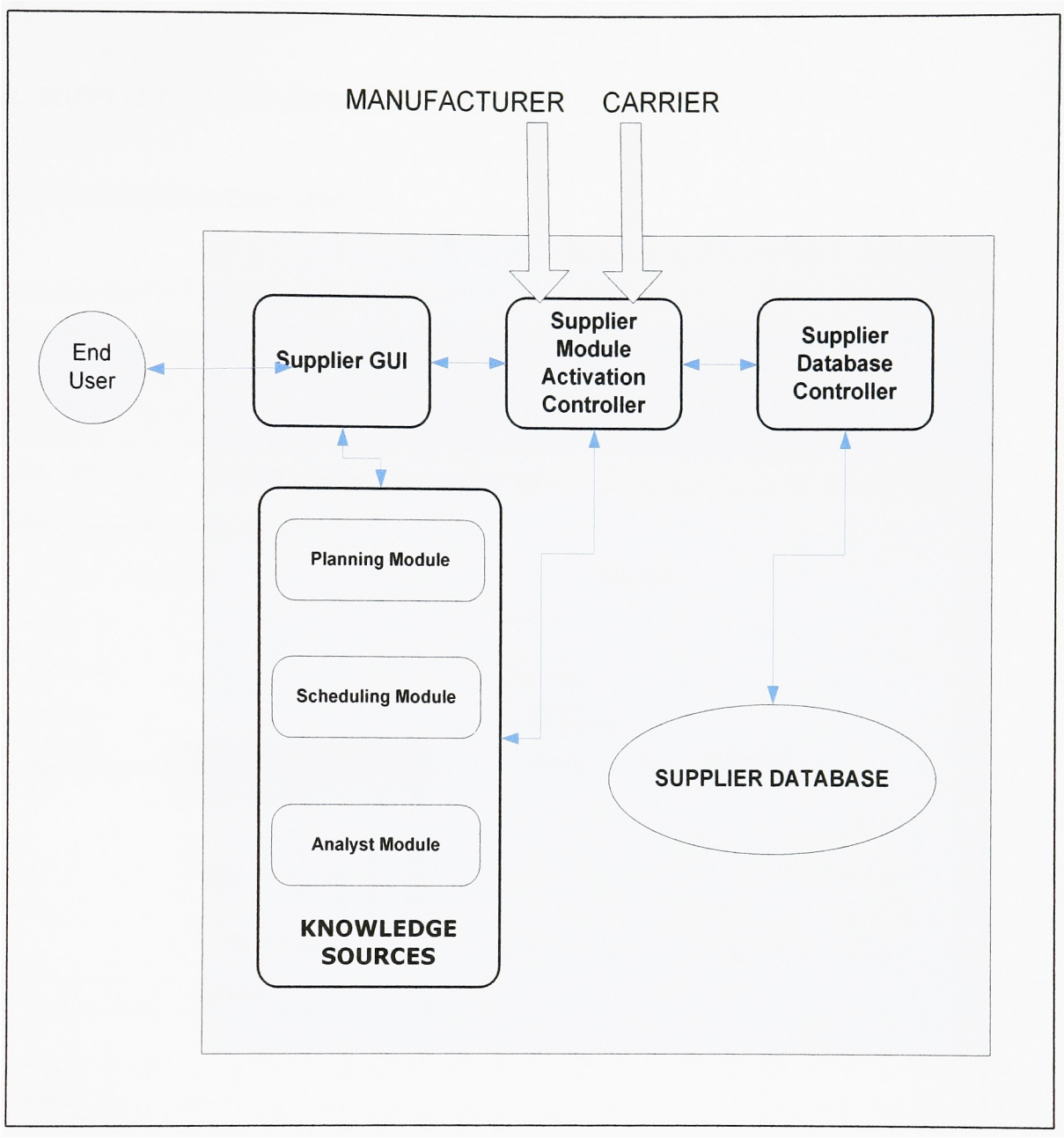


Figure 7.5: Block Diagram for Supplier

7.2.4.1. SUPPLIER AGENT MODULES

➢ **Supplier Graphical User Interface:**

Figure shows the GUI for the supplier.

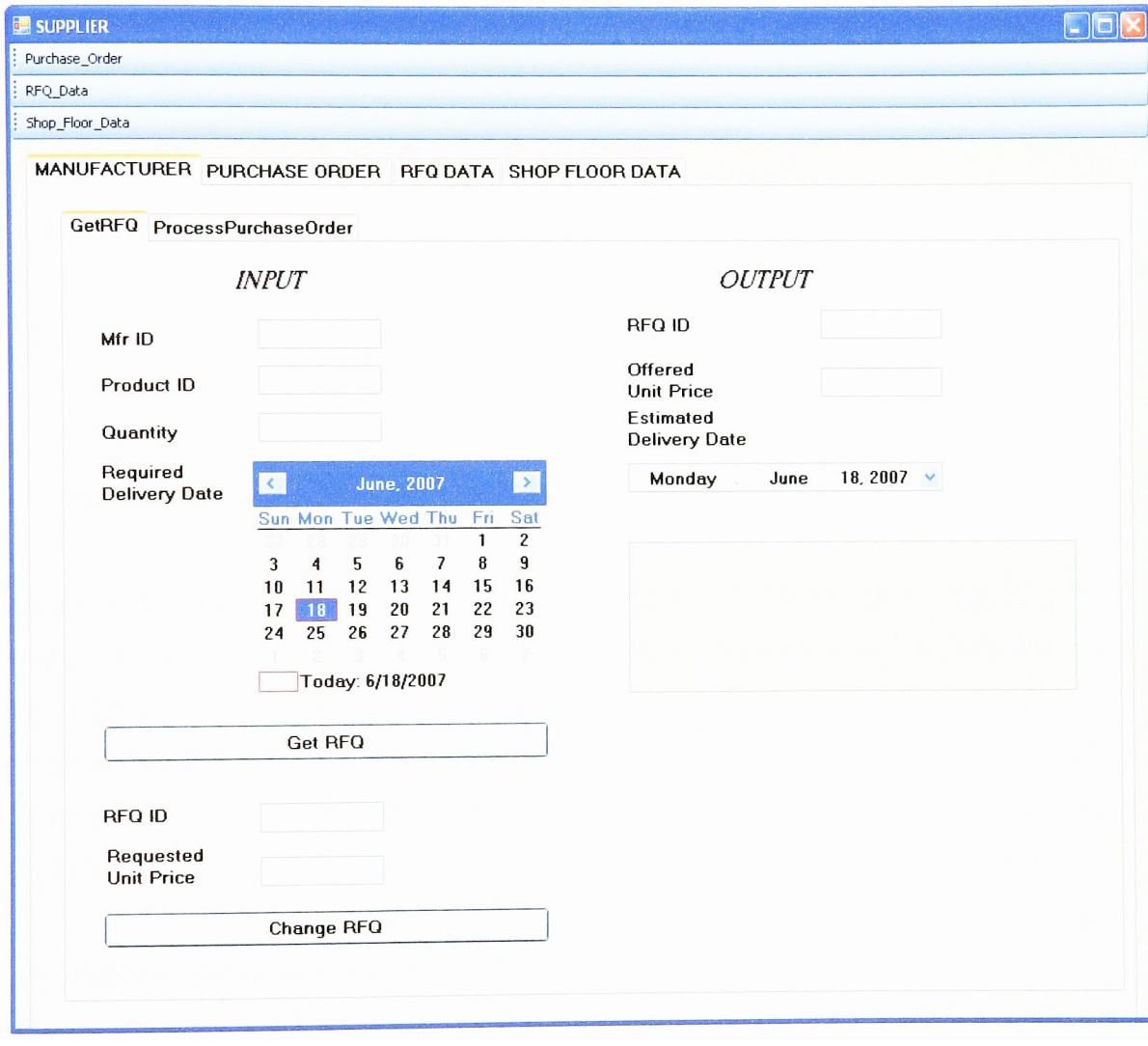


Figure 7.6: Graphical User Interface for Supplier

The interaction of a manufacturer with the supplier can be simulated using this GUI. The manufacturer can send a "GetRFQ" BOD to the supplier to request quote information for a product. Supplier responds to this request by sending a "ShowRFQ" BOD. A manufacturer

can also negotiate the price offered in a "ShowRFQ" response using a "ChangeRFQ" BOD. Once the manufacturer is satisfied with the RFQ it can do purchase order interactions.

A complete list of all the interactions between a manufacturer and a supplier which can be simulated using this implementation are as follows

GetPurchaseOrder ↔ ShowPurchaseOrder,

ProcessPurchaseOrder ↔ AcknowledgePurchaseOrder,

ChangePurchaseOrder ↔ AcknowledgePurchaseOrder and

CancelPurchaseOrder ↔ AcknowledgePurchaseOrder interaction.

PURCHASEORDER, RFQ DATA and SHOP FLOOR DATA tabs are used to display relevant information from the supplier database.

The carrier does not initiate any interaction with the supplier and hence the user cannot simulate carrier interaction with the supplier using SGUI.

- **Supplier Module Activation Controller (SMAC)**

Supplier Module Activation Controller receives all the BODs which are sent from the GUI screen. This is equivalent to receiving the BODs from manufacturers. It directs the control to the correct decision making method in the knowledge source module of the supplier. It also supplies the knowledge source module with all the information it will need to make the necessary decision. The SMAC retrieves all this information using the supplier database controller module. SMAC then convey the decisions made by SKS back to the manufacturer.

- **Supplier Database Controller (SDC)**

The supplier database controller module is responsible for all the interactions with the supplier database. It retrieves all the information required by SMAC from the database and also updates records. The SDC is the only module which is allowed to interact with the database. No other module in the agent is allowed any interaction with the database.

- **Supplier Knowledge Source (SKS)**

The knowledge source module of the supplier is where all the decisions for the supplier are taken. SMAC sends a request to SKS and asks it to take a decision. It also provides SKS with all the relevant information it will need to perform the operation of decision making. SKS then returns the result back to SMAC which is then used to populate a response to a manufacturer.

- **Supplier Database**

Microsoft SQL Server 2005 Express is used to create the database architecture for the supplier. Supplier stores the data in this database.

7.2.5. INTERACTION DETAILS FOR SUPPLIER AGENT

Manufacturer interacts with the supplier. The following section explains all the interactions between them.

7.2.5.1. Request for quote

A manufacturer would like to know at what price the supplier is offering a particular product. For this purpose a manufacturer uses a "GetRFQ" BOD to request a quote from the supplier.

Table 7.8: GetRFQ interaction between Supplier and Manufacturer

No.	Manufacturer	Supplier	Decisions taken
8.1	GetRFQ The manufacturer sends this message to place a request for quote for a particular item. Manufacturer also sends the quantity required and the delivery date on which it would like to receive the shipment if he places an order for it.		
8.2		ShowRFQ SMAC receives GetRFQ from manufacturer. SMKS then decides if it should continue interaction with the manufacturer. If interaction is to be continued then SMKS decides the price to quote for the requested item and quantity. It also checks if desired delivery date can be meet using the current inventory in stock or if it can be manufactured before the requested delivery date. If it cannot be manufactured by the requested delivery date then what is the earliest possible delivery date	➢ Should we continue negotiation with the retailer? ➢ Is there is enough inventory to complete the requested order? ➢ What is the earliest possible delivery date by which we can meet the

		by which the requested order can be meet. If negotiation is not to be continued then the manufacturer is informed accordingly. This transaction information is recorded in its database.	order?

The manufacturer would like to request a change in the price offered by the supplier. For this the manufacturer uses the "ChangeRFQ" message.

Table 7.9: ChangeRFQ interaction between Supplier and Manufacturer

No.	Manufacturer	Supplier	Decisions taken
9.1	ChangeRFQ The manufacturer uses the RFQ number it received during GetRFQ transaction from the supplier to place a request to change the price offered by the supplier for a product.		
9.2		AcknowledgeRFQ The supplier receives the ChangeRFQ request from manufacturer. The supplier then checks the status of the manufacturer and based on the status information it offers a new price to the manufacturer. It records this information in his database.	➢ What new price to offer?

7.2.5.2 Purchase Order

When a manufacturer wants to place an order for goods from the supplier, it uses a "GetPurchaseOrder" BOD. In this application the manufacturer needs to have an RFQ number as a reference to proceed with purchase order transactions.

Table 7.10: GetPurchaseOrder interaction between Supplier and Manufacturer

No	Manufacturer	Supplier	Decisions taken
10.1	GetPurchaseOrder The manufacturer initiates the process of placing an order by sending this message to the manufacturer. Manufacturer sends the RFQ number it received for the product during RFQ transaction		
10.2		ShowPurchaseOrder The supplier retrieves the RFQ number from the GetRFQ message and retrieves all the relevant information from its database related to the RFQ transaction and sends this information back to the manufacturer using ShowPurchaseOrder message.	

Now, the manufacturer can go ahead and place the order using a "ProcessPurchaseOrder" message. If however the manufacturer wants to change the purchase order he can use the

"ChangePurchaseOrder" message. The manufacturer cannot change a purchase order once it has been processed and agreed upon using the "ProcessPurchaseOrder" message.

Table 7.11: ChangePurchaseOrder interaction between Supplier and Manufacturer

No	Manufacturer	Supplier	Decisions taken
11.1	ChangePurchaseOrder If the manufacturer wants to change the quantity desired or the delivery date required, then he can use this message.		
11.2		AcknowledgePurchaseOrder The supplier then decides if it can allow the change requested by the manufacturer and conveys the result to the retailer. It records the changes in its database.	➢ Allow the change? ➢ Is there is enough inventory to complete the requested order? ➢ What is the earliest possible delivery date by which we can meet the order?

Table 7.12: ProcessPurchaseOrder interaction between Supplier and Manufacturer

No	Manufacturer	Supplier	Decisions taken
12.1	ProcessPurchaseOrder The manufacturer uses this message to place the order		
12.2		AcknowledgePurchaseOrder The supplier accepts the request to process the order and records this information in its database.	

Once the order is placed, if the manufacturer decides to cancel the order then it can try to do so using the CancelPurchaseOrder message.

Table 7.13: CancelPurchaseOrder interaction between Supplier and Manufacturer

No	Manufacturer	Supplier	Decisions taken
13.1	CancelPurchaseOrder Manufacturer uses this message to try and cancel the purchase order		
13.2		AcknowledgePurchaseOrder The SMAC receives this request and using the SMKS decides what penalty to charge the manufacturer for canceling the order.	➢ What should be the penalty?

7.2.6. DECISIONS TAKEN IN A SUPPLIER AGENT

The decision making ability in the supplier agent is present in the supplier knowledge source module. In this thesis, the supplier knowledge source module can perform the following decisions

- Should we continue negotiation with the manufacturer?
- Decide the unit price to be offered.
- Decide the earliest possible delivery date.
- New updated price to be offered.
- Calculate the penalty for cancellation of a purchase order.

Should we continue negotiation with the manufacturer?

When a manufacturer places a "GetRFQ" message, the SMAC receives it and asks the SMKS to decide if the supplier should continue negotiation with the manufacturer. If the status is "Poor" then further negotiation is refused else further negotiation is initiated.

Decide the unit price to be offered?

This decision is taken when a manufacturer places a request for quote for a particular item. The input factors that are considered while making this decision are the current inventory level of the product, the required quantity, the quantity ranges and the corresponding unit price which the supplier has set beforehand and a boolean value which indicates if the required quantity can be manufactured. First the required quantity is compared against the quantity ranges and the corresponding price is set as the unit price. Then a check of the required quantity is done against the current inventory level of the product and if the required quantity is less than the current inventory level then the set unit price is returned as result. However, if the required quantity is greater than the current inventory level and requires production of the product, then the unit price is increased by an amount equal to 15% of the previous unit price.

Decide the earliest possible delivery date?

Whenever the requested delivery date cannot be meet, this decision is required to be made to calculate the earliest possible delivery date when the order can be meet. The inputs to this method are the inline current production, the capacity of the shop floor to manufacture the product in a production day, the requested quantity and the number of days the carrier would take to make the delivery. The number of days the carrier would take to make the delivery is an arbitrary number and can be changed. It is assumed, for simplicity, that only a single machine is available to make a particular product. The formula used is as follows

Possible Return Date = ((linedupprod/sum) + (quantity/sum) + carrierdays);

Where linedupproduction – current inline production

sum = capacity of the shop floor to manufacture the product in a production day,

quantity = requested quantity and

carrierdays = number of days the carrier would take to make the delivery.

New updated price to be offered?

This decision is taken in response to a request from a manufacturer to reduce the offered price. The offered price is shown to the manufacturer through the ShowRFQ BOD. If the manufacturer is not satisfied with the price offered to him, he/she can request a reduction in the price. There are two input factors to this method, the status of the manufacturer and the offered old price. The status of the manufacturer is based on his past record with the supplier. There are four levels to the status namely

New – if the manufacturer has no past history with the supplier and is interacting with the supplier for the first time.

Good – manufacturer has a good record and has not cancelled more then 3 purchase orders

Very Good – If the manufacturer has not cancelled any purchase orders he has placed with the supplier.

Poor – If the manufacturer has cancelled more then 3 purchase orders

Based on these status levels, a decision to update the price is taken. If the retailer status is very good, then a 10% reduction in the old price is offered. If the manufacturer status is

"Good" or "New" then a 5% reduction is offered. If his status is Poor, then no reduction is offered.

Calculate the penalty for cancellation of a purchase order?

If a manufacturer wishes to cancel a purchase order he/she has placed then this decision is made. The inputs to this decision are the promised delivery date and the offered unit price. The penalty calculation is based on the difference between the promised delivery date and the current date. If the difference is less than seven days then the manufacturer is not allowed to cancel his/her purchase order. If the difference is between seven to fourteen days then the manufacturer is charged a penalty of 30% of the unit price. And if the difference between the promised delivery date and the current date is more than fourteen days then the manufacturer can cancel his/ her purchase order without incurring any penalty.

7.2.7. CARRIER SYSTEM

The carrier system is responsible for transporting raw materials from supplier to manufacturers and also finished product from manufacturers to retailers. There is interaction between the manufacturer and the carrier, between the supplier and the carrier and also between the retailer and the carrier. Similar to the manufacturer and the supplier system, the carrier system also consists of the following basic modules.

- Carrier Graphical User Interface (CGUI)
- Carrier Module Activation Controller (CMAC)
- Carrier Database Controller (CDC)
- Carrier Knowledge Source (CKS) and
- Carrier Database

Block Diagram of the carrier system is as shown in Figure 7.7. In this thesis implementation, the carrier system has two modes of transportation, namely Road and Air. For purposes of distance calculations it is assumed that the carrier is centered at New York City, New York and is directly connected to eight other states as shown in table 7.14. If the delivery is required to be made to one of the eight states then the time calculation are considered accordingly. If delivery to any state other then the eight states is required then an average of three days is assumed to be required from the day the required capacity will become available.

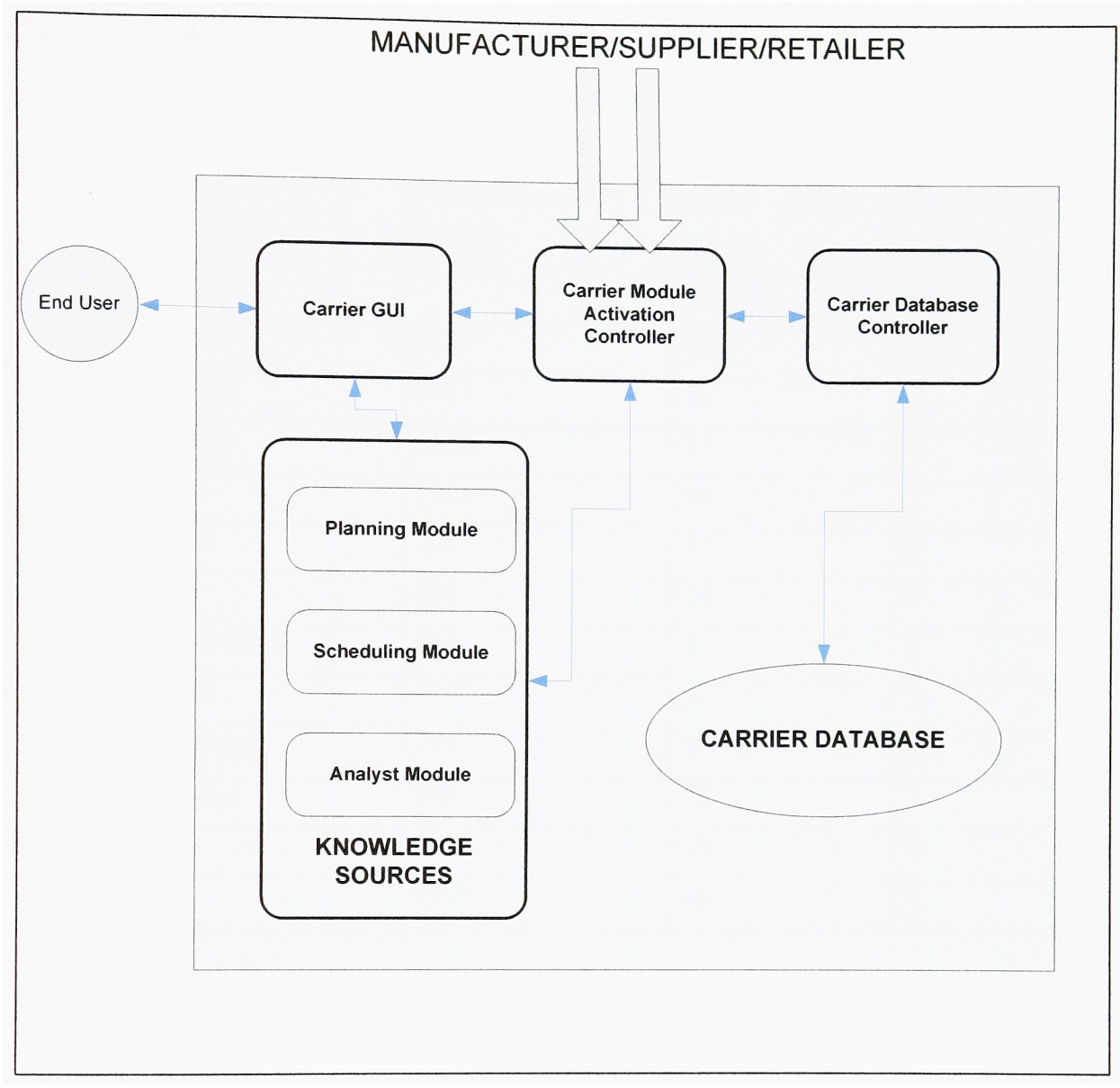


Figure 7.7: Carrier Block Diagram

Table 7.14: Base states information for the carrier system (http://www.abo.fi/~oholm/distance/USdistance.shtml)

No.	State	Total Distance (miles)	Total time – Road (hours)	Total time – Air (hours)
1	Alaska	4514	106	9
2	California	2911	59	6
3	Colorado	1852	37	4
4	Florida	1325	26	3
5	Georgia	863	17	2
6	Illinois	818	16	2
7	Massachusetts	216	4	1
8	Nevada	4170	52	5
9	Pennsylvania	376	8	1
10	Tennessee	949	19	2
11	Texas	1649	33	3
12	Utah	2275	46	5

The following section describes the carrier system modules in detail.

7.2.7.1. CARRIER AGENT MODULES

- **Carrier Graphical User Interface (CGUI)**

The carrier graphical interface is as shown in figure:

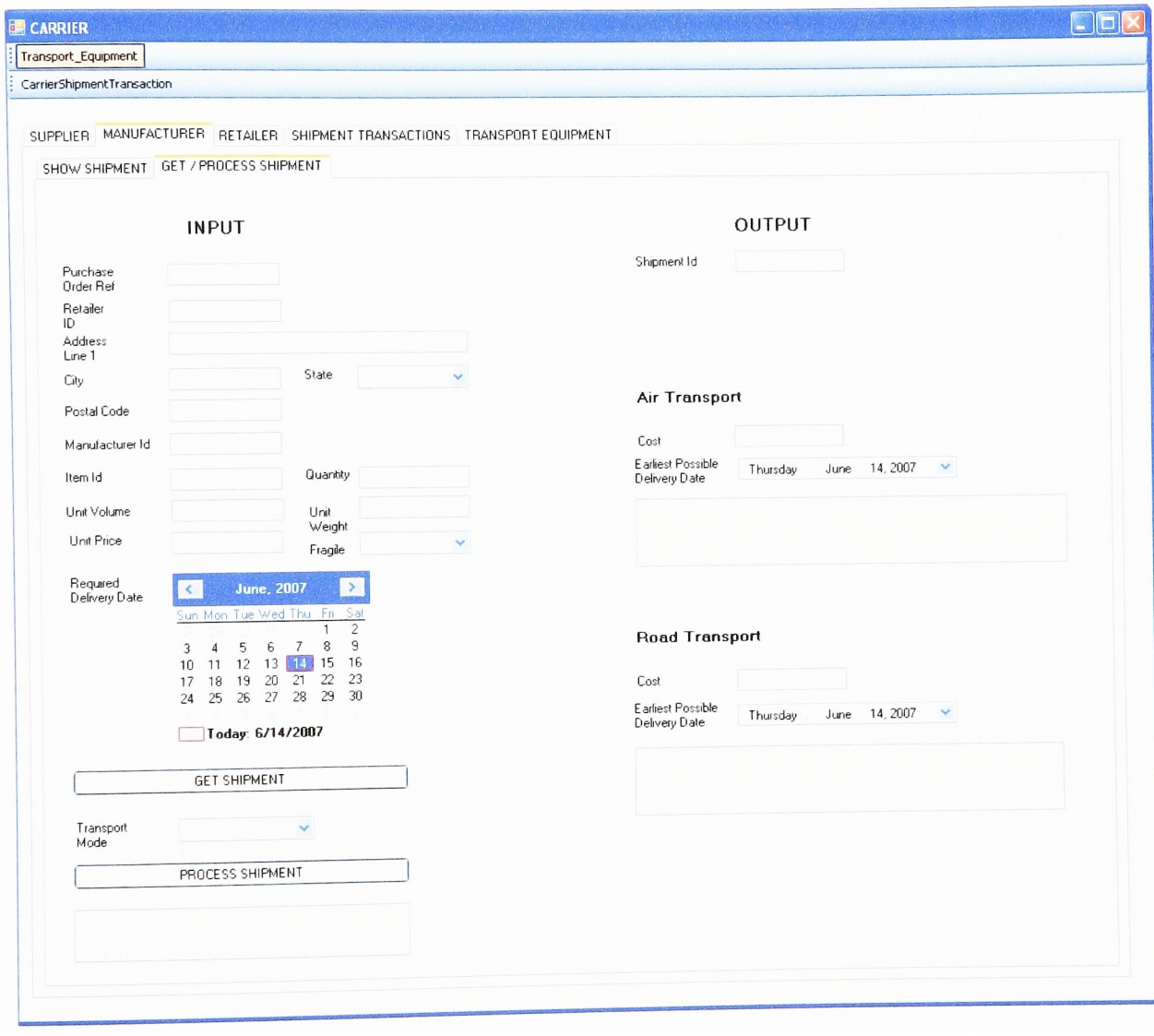


Figure 7.8: Carrier Graphical User Interface

This CGUI simulates the interaction of manufacturer, supplier and retailer with the carrier module. To ship his order to a retailer, the manufacture initiates interaction with the carrier agent by sending a "GetShipment" message. The carrier responds to this by sending back a

"ShowShipment" message. Now, if the manufacturer wants he can accept the offered shipment by placing a "ProcessShipment" message. To this the carrier responds by sending back "AcknowledgeShipment" message. This scenario is also possible for a supplier who wished to ship his consignment to a manufacturer. Manufacturer and retailer can also send a "GetShipment" message to the carrier to know the status of their shipment.

- **Carrier Module Activation Controller (CMAC)**

Carrier Module Activation Controller receives all the BODs which are sent from the CGUI screen which is equivalent to receiving the BODs from manufacturers, supplier or retailers. It then directs the control to the correct decision making method in its knowledge source module. It also supplies the knowledge source module with all the information it will need to make the necessary decision. The CMAC retrieves all this information using the carrier database controller module. CMAC then convey the decisions made by CKS back to the interacting partner.

- **Carrier Database Controller (CDC)**

The carrier database controller module is responsible for all the interactions with the carrier database. It retrieves all the information required by CMAC from the database and also updates records. The CDC is the only module which is allowed to interact with the carrier database. No other module in the agent is allowed to interact with the database.

- **Carrier Knowledge Source (CKS)**

The knowledge source module of the carrier is where all the decisions for the carrier are taken. CMAC sends a request to CKS and asks it to take a decision. It also provides CKS with all the relevant information it will need to perform the operation of decision making. CKS then returns the result back to CMAC which is then used to populate a response to a manufacturer.

➢ **Carrier Database**

Microsoft SQL Server 2005 Express is used to create the database architecture for the carrier. Carrier stores the data in this database

7.2.8. INTERACTION DETAILS FOR CARRIER

The following table present information related to all the interactions with the carrier. It also shows all the decisions that are made during a particular interaction

7.2.8.1. Shipment

Table 7.15: GetShipment interaction between the carrier and manufacturer/supplier/retailer

No.	Manufacturer/Supplier/Retailer	Carrier	Decisions taken
15.1	GetShipment Manufactucturer/Supplier uses this BOD to request a quote for shipment. They send information related to the destination of delivery, the product delivered, the order quantity, required delivery date, brittleness of the product to be delivered, volume and weight of a unit product, manufacturer/supplier IDs and the related purchase order number. A manufacturer or retailer can also use this BOD to get an update on their shipments.		

15.2		ShowShipment The carrier responds to this by sending back a ShowShipment BOD. Carrier sends back information related to the cost and delivery date associated with both road and air transport. This information is recorded in the database.	➢ What price to offer for Road and Air transport ➢ What delivery date is possible for Road and Air transport.

Now if the manufacturer/supplier is satisfied with the "ShowShipment" message, he can request the carrier to process shipment by sending a "ProcessShipment" BOD.

Table 7.16: ProcessShipment interaction between the carrier and manufacturer/supplier

No.	Manufacturer/Supplier	Carrier	Decisions taken
16.1	ProcessShipment This BOD is used to request the carrier to place a shipment order. The manufacturer/supplier selects the mode of transport that it would like the carrier to use. It also send the Shipment number which it received during the GetShipment transaction.		
16.2		AcknowledgeShipment Carrier acknowledges the	

		request by sending back an AcknowledgeShipment BOD. The shipment status is then updated in the database.	

7.2.9. DECISION TAKEN IN THE CARRIER AGENT

The carrier knowledge source (CKS) is responsible for performing the task of decision making.

Carrier module activation controller (CMAC) initiates the decision making process by sending a request to CKS and also provides it with all the necessary information that it needs to perform the decision. There are two important decisions taken by the CKS, namely

- What price to offer for Air and Road Transport and
- What delivery date is possible for Air and Road Transport?

What Price to offer for Air and Road Transport?

This decision is required to calculate what price to offer for Road and Air transport. The inputs to this decision are as follows:

- Boolean value indicating if the total volume required for making the shipment is currently available.
- The status of the manufacturer/supplier.
- The price to transport a unit of the product by either air or road.
- The required quantity to be shipped.
- Boolean value indicating if the product to be shipped is fragile and

If the current capacity available to the carrier is more than the required quantity to be delivered, then the boolean value indicating the total volume is true. The status of the manufacturer can either be "Very Good", "Good" or "Poor" depending on his past record with the carrier. The price to transport a unit of the product is based on the total volume and weight of the shipment to transport. This value is retrieved from the carrier database. The required

quantity to be shipped and the boolean value indicating if the product to be shipped is fragile are provided by the manufacturer/supplier, which is simulated through CGUI.

Initially the price to be returned is set equal to the unit price value. If the product is fragile then there is a 10% increase in the unit price. Also if the total volume required for transportation is currently available then the price is further increased by 10%. However, if the total volume is not available then there is a 5% decrease in the unit price. Now the status of the manufacturer/supplier is checked and if it is "New" or "Good" then a further 10% discount is offered on the unit price. If the status is "Very Good" then a discount of 15% is offered. If the status of manufacturer/supplier is "Poor" then no further discount is offered. Finally the unit price is multiplied by the quantity of product to be transmitted to return the final price.
This decision is called separately for Air and Road transport. The unit price parameter value is based on the mode of transport value which is decided by the CMAC.

What delivery date is possible for Road and Air transport?
If the destination where the delivery is required is in one of the eight states, then the value of the time required to delivery by road or air is retrieved from the database. However, if the destination is not one of the eight states then it is assumed that a total of three days is required for shipment by road and air. If the total capacity volume required for making the shipment is available then the delivery date is equivalent to the number of days required for delivery. If the total capacity is not available then the delivery date is equal to the number of days required for delivery plus the number of days it will take for the total capacity to be available to make the shipment. Finally a total of seven days for road and three days for air transport are added as a buffer time to allow the manufacturer/supplier to responds the "ShowShipment" message. If the manufacturer/supplier does not respond to the "ShowShipment" message in affirmative within the allotted time frame then the shipment cannot be made as promised in the "GetShipment" message.

8. DEVELOPMENT OF SCM SUPPORT DATABASES

This section presents the work done in designing SCM databases for this thesis. It provides an introduction to Microsoft SQL Server 2005 Express and presents the different databases developed for supplier, manufacturer and carrier.

8.1. MICROSOFT SQL SERVER 2005 EXPRESS (George)

SQL server 2005 Express is a free and easy to use database product that is based on SQL server 2005 technology. It is designed to provide a database platform that offers superior ease of use, enabling fast deployments for its target scenarios. It includes different free tools like SQL Server Management Studio Express Edition, Surface Area Configuration Tool, and SQL server configuration Manager, to simplify the basic database operations. It also has the ability to integrate with Visual Studio projects.

8.2. DATABASE DESIGN

The projects uses separate data bases for each agent, the three databases are manufacturer database, supplier database, and carrier database. These databases are in accordance with the real life data stored in industry applications. These databases store the information that is exchanged between the agents through xml files. The different tables in the thesis are based on the Oagis business object documents used to construct the different scenarios presented in the thesis.

The databases are designed on the principles of normalization. A third level of normalization is used in database to eliminate redundant data and ensure the required data dependencies.

The following section describes each database in detail. These are not actual database patterns used in industry but they cater to the requirements of this project.

8.2.1. MANUFACTURER DATABASE

This database serves the manufacturer requirements of storing the purchase orders, RFQs, shipment data and so on. The database tables resemble the various functionality needs. For instance the table 'ManufacturerPurchaseOrder Data' stores the information incoming from a typical purchase order placed with manufacturer. This design enables independent execution

of different functionalities. The various tables for manufacturer database and the corresponding functionality that it serves are as follows:

1. ManufacturerPurchaseOrderData
2. ManufacturerRFQData
3. ManufacturerOrderedProductData
4. ManufacturerShopFloorData
5. ManufacturerRetailer
6. ManufacturerCarrier
7. ManufacturerSupplier
8. ManufacturerInvoiceData
9. ManufacturerBillOfMaterial
10. ManufacturerPriceDistribution
11. InventoryOfManufacturer
12. SupplierInventory
13. ManufacturerShipmentData

'ManufacturerPurchaseOrderData' table is used to store information related to different purchase orders placed or in process with the manufacturer. 'PODocumentID' field in the table is used to store or retrieve any data related to a particular purchase order. Similarly any data related to RFQ transactions can be processed by using the 'RFQDocumentId' field in the 'ManufacturerRFQData' table which is used to store data related to RFQ transactions. 'ManufacturerOrderedProductData' table stores information about different products ordered from the manufacturer and can be used to plan production. Production planning data is stored in the 'ManufacturerShopFloorData' table which shoes the load on individual machines on the manufacturers shop floor. The 'ManufacturerRetailer','ManufacturerCarrier', and 'ManufacturerSupplier' tables are used to stores contact details of different retailers, carriers and suppliers respectively with whom the manufacturer interacts. ' ManufacturerBillOfMaterial' table stores Bill Of Material data. The pricing details of different products are stored in the 'ManufacturerPriceDistribution' table. These pricing details are used while deciding the price of a product to be offered to customer during RFQ

transactions. 'InventoryOfManufacturer' table keeps a record of manufacturers present inventory level. 'SupplierInventory' table keeps a record of all the inventory received from different suppliers. Manufacturers shipment details are recorded in the 'ManufacturerShipmentData' table.

8.2.2. SUPPLIER DATABASE

This database is used to store supplier data. Data related to different purchase orders, RFQ's, shipment data is stored in this database. Its design is similar to manufacturer agent database and it comprises of the following tables

1. SupplierPurchaseOrderData
2. SupplierRFQData
3. SupplierShopFloorData
4. SupplierOrderedProductData
5. SupplierCarrier
6. SupplierManufacturer
7. SupplierInvoiceData
8. SupplierBillOfMaterial
9. SupplierShipmentData
10. SupplierPriceDistribution
11. ManufacturerInventory
12. InventoryOfSupplier

The tables perform similar functions as explained for the manufacturer agent database, only 'ManufacturerInventory' table can be used to store information about the current inventory levels of different manufacturers and can be used to plan suppliers production accordingly.

8.2.3. CARRIER DATABASE

This database is used to store information of the carrier agent. Different information like ProcessShipment data, transport equipment data, price distribution data is stored in the table of this database. Different tables of this database are as follows

1. CarrierShipmentTransaction
2. CarrierRetailerTable
3. CarrierSupplier
4. CarrierManufacturer
5. CarrierTransportEquipment
6. States
7. CarrierPrice

The 'CarrierShipmentTransaction' table stores information related to all the different shipment orders that are placed or in negotiation with the carrier. 'CarrierRetailerTable', 'CarrierSupplier', and 'CarrierManufacturer' tables store information about the retailers, suppliers and manufacturers with whom the carrier interacts. 'CarrierTransportEquipment' table stores information about different transport equipments available with the carrier, their capacity, whether they are currently available to carry new loads or if they are busy at what earliest date they will be available. 'States' table is used to store information about all the different states to which the carrier can make direct delivery or those states in which the carrier has base stations. 'CarrierPrice' table is used to store price information. Air and road transport price information to different base states is stored in this table.

The detailed description of all the tables for each of the databases and their internal relationships can be found in Appendix 4 .

9. IMPLEMENTATION DETAILS

In this architecture, the manufacturer is represented by a single agent based on the modified MASCOT agent architecture. Supplier and the carrier systems each are similarly represented by agents. The agent comprises a graphical user interface, a module activation controller, a database controller module, a knowledge source module and a database. Two agents interact with each other using their respective module activation controller web service. As the graphical user interface for a system simulates the interaction of other agents with its corresponding agent, it does the work of a module activation controller for other agents interacting with the module activation controller of the corresponding agent.

The complete cycle of interaction between two agents and the decision making process involved can be explained with the help of the following example. Suppose, a manufacturer places a request for quote with the supplier agent for a product, the following events occur

1) A request is generated from the manufacturer tab in the supplier graphical user interface. This request is in the form of an XML document based on the GetRFQ Business Object Document from OAGIS. The XML document is as shown in figure 9.1:

```
<?xml version="1.0" encoding="UTF-8"?>

<GetRFQ xmlns="http://www.openapplications.org/oagis/9"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openapplications.org/oagis/9
../../BODs/Developer/GetRFQ.xsd" languageCode="en-US"
versionID="normalizedString" releaseID="normalizedString"
systemEnvironmentCode="Production">
    <DataArea>
        <Get uniqueIndicator="false" recordSetStartNumber="2"
            recordSetSaveIndicator="false" maxItems="2"
            recordSetReferenceId="normalizedString">
            <Expression expressionLanguage="token">token</Expression>
        </Get>
        <RFQ>
            <RFQLine>
                <Quantity unitCode="normalizedString">2000</Quantity>
                <RequiredDeliveryDateTime>2007-12-
                  07</RequiredDeliveryDateTime>
                <RequesterParty category="Organization">
                    <PartyIDs>
                        <ID schemeAgencyName="String"
                                schemeAgencyID="normalizedString"
                                schemeVersionID="normalizedString"
                                schemeName="String"
                                schemeURI="http://www.openapplications.org"
                                schemeID="normalizedString"
                                schemeDataURI="http://www.openapplications.or
                                g">12345</ID>
                    </PartyIDs>
                </RequesterParty>
                <Item>
                    <ItemID agencyRole="Text">
                        <ID schemeAgencyName="String"
                        schemeAgencyID="normalizedString"
                        schemeVersionID="normalizedString"
                        schemeName="String"
```

```
                    schemeURI="http://www.openapplications.org"
                    schemeID="normalizedString"
                    schemeDataURI="http://www.openapplications.org">123
                    </ID>
                </ItemID>
            </Item>
        </RFQLine>
    </RFQ>
</DataArea>
</GetRFQ>
```

Figure 9.1:Example XML payload message based on GetRFQ BOD

1) This XML document is then wrapped into a SOAP message, which is delivered over the internet to the module activation controller web service of the supplier agent.
2) Supplier agent module activation controller web service then receives this SOAP message and retrieves the OAGIS payload.
3) The web service then parses the XML document to retrieve the relevant fields. E.g. manufacturer Id, requested delivery date, product Id, requested quantity.
4) The module activation controller then executes a sequence of events to prepare a ShowRFQ BOD which is explained as follows
    a. MAC then requests the database controller web service of the supplier agent to record the request in its database.
    b. MAC then retrieves the current inventory level and the pre set price details for the requested product.
    c. If the available quantity is more then the requested quantity then the MAC requests the knowledge source web service for supplier to decide the unit price to offer, and sets the possible delivery date as the requested delivery date.
    d. If the available quantity is less then the requested quantity, MAC retrieves the current inline production and the current production capacity of the supplier shop floor for the requested product. If the current shop floor parameters enable production of the requested quantity by the required delivery date, then

MAC updates the current inline production for the requested product and requests the knowledge source web service to calculate the price.

e. If the requested product cannot be produced by the requested delivery date, MAC requests knowledge source web service to calculate the earliest possible delivery date, when it would be possible to complete the order.

5) MAC then populates this data into a ShowRFQ XML document which is then returned to the manufacturer agent, here it is returned to the manufacturer tab on the supplier graphical user interface. It also request the database controller to record the promised due date and the offered price in its database.

In this thesis, since the main focus is on using agent technology for supply chain management, we are mainly concerned with how the agent responds to a request from other agent. As explained in the above example, when the supplier agent receives a request, the module activation controller for that agent retrieves different data values from the XML document. It then performs a serious of checks to decide the future course of action to be taken to create a response to the request. It uses the database controller web service to retrieve and update information to the database and also uses the knowledge source web service to make relevant decisions.

Creation of this application involves building a user interface, three web services and a database for each system agent. The implementation of each agent is done using Microsoft Visual Studio 2005. Each module in an agent is created as a separate project. Microsoft Windows project is used to implement the graphical user interface, while the remaining modules are implemented using ASP.Net Web Service.

The steps to create such an application are explained in the following pages.

You must have Microsoft Visual Studio 2005 installed on your machine.

Converting XML schemas used in this application into directly usable C# classes:

1) Download the OAGIS schemas. OAGIS 9.0 schemas were used in this application.
2) Download the XSDObjectGen tool from Microsoft website and add it as a tool to visual studio.

3) Take one schema and apply the XSDObjectGen tool to that schema. The tool generates one main file and seven supporting C# class files. Give a namespace to each of the file. In this thesis namespace "Vilesh_Thesis" was used for the main class file.
4) Repeat step three for all the XML schemas to be used in the application.
5) Make the following changes in the main file associated with each XML schema.
   - Change the ("bool") data type to ("Boolean").
     a. E.g.

Before Change:

```
[XmlAttribute(AttributeName = "inline", Form =
XmlSchemaForm.Unqualified, DataType = "bool", Namespace =
Declarations.SchemaVersion)]
```

After Change:

```
[XmlAttribute(AttributeName = "inline", Form =
XmlSchemaForm.Unqualified, DataType = "boolean", Namespace =
Declarations.SchemaVersion)]
```

Change the (byte[]) data type to (base64Binary).

b. E.g.

Before Change:

```
[XmlText(DataType = "byte[]")]
```

After Change:

```
[XmlText(DataType = "base64Binary")]
```

Change the datatype (object) to (string)

c. E.g.

Before Change:

```
[XmlAttribute(AttributeName = "listAgencyName", Form =
XmlSchemaForm.Unqualified, DataType = "object", Namespace =
Declarations.SchemaVersion)]
[EditorBrowsable(EditorBrowsableState.Advanced)]
public object __listAgencyName;

[XmlIgnore]
public object listAgencyName
{
    get { return __listAgencyName; }
    set { __listAgencyName = value; }
   }
```

After Change:

```
[XmlAttribute(AttributeName = "listAgencyName", Form =
XmlSchemaForm.Unqualified, DataType = "string", Namespace =
Declarations.SchemaVersion)]
[EditorBrowsable(EditorBrowsableState.Advanced)]
public string __listAgencyName;

[XmlIgnore]
public string listAgencyName
{
    get { return __listAgencyName; }
    set { __listAgencyName = value; }
}
```

These changes are required to make the C# class files generated by the XSDObjectGEN tool compatible with ASP.Net web services.

6) Different OAGIS 9.0 schemas have various common elements. The main C# class files generated contain classes which are common to each other. To be able to use the C# class file in this application, create a single main class file which contains all the classes only once. Repetition of classes in the main class file is not permitted.

The following section describes the procedure to create a web service.

Creating a Web service:

1. For creating and operating web service using Visual Studio 2005, you will need the IIS (Internet Information Service) installed.
2. Create a new ASP.Net web service project. The project will, by default, have a Service.asmx and App_Code\Service.cs files. The Service.cs file has a class called "Service", which is our Web Service. Any methods of this class with the attribute [WebMethod} before it will become an operation of the class. This file by default has the following content:

```
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    public Service () {
        //Uncomment the following line if using designe components
        //InitializeComponent();
    }
    [WebMethod]
    public string HelloWorld() {
        return "Hello World";
    }
}
```

Figure 9.2: Defaul Service.cs file content

Now, to be able to use our schemas, we need to add the main C# class file, with no repeated classes, and the associated files generated by the XSDObjectGen tool to the App_Code folder of this project. Add the following line at the top of the file

"using namespace_name;"

where the namespace_name is the name of the namespace associated with the main class file. To create the applications supported by this Web Service, we need to create public methods of the Service class with the attribute [WebMethod].

As explained in the example above, when a manufacturer places a GetRFQ request with the supplier, a method with code shown in Fig 8.3 is executed at the graphical user interface to generate the XML document.

```
private void btn_GetRFQ_Click(object sender, EventArgs e)
        {

            //generate the GetRFQ object

            GetRFQ grfq = new GetRFQ();
            //generate the RFQ ID
            //grfq.ApplicationArea = new ApplicationArea();
            //grfq.ApplicationArea.BODID = new BODID();
            //grfq.ApplicationArea.BODID.Value = "RFQ" + (new
Random()).Next(100000).ToString();

            //populate the quantity
            grfq.DataArea = new GetRFQDataAreaType();
            grfq.DataArea.RFQ = new RFQ[1];
            grfq.DataArea.RFQ[0] = new RFQ();
            grfq.DataArea.RFQ[0].RFQLine = new RFQLine[1];
            grfq.DataArea.RFQ[0].RFQLine[0] = new RFQLine();
            grfq.DataArea.RFQ[0].RFQLine[0].Quantity = new Quantity();
            grfq.DataArea.RFQ[0].RFQLine[0].Quantity.Value =
Decimal.Parse(txt_Quantity.Text);

            //add the requested delivery date
            grfq.DataArea.RFQ[0].RFQLine[0].RequiredDeliveryDateTime =
monthCalendar1.SelectionStart.ToShortDateString();

            //add the retailer id
            grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty = new
RequesterParty();
            grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty.PartyIDs = new
PartyIDs();
            grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty.PartyIDs.ID =
new ID[1];
            grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty.PartyIDs.ID[0] =
new ID();
```

```
grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty.PartyIDs.ID[0].Value =
txt_MfrId.Text;

            //add the product id
            grfq.DataArea.RFQ[0].RFQLine[0].Item = new Item();
            grfq.DataArea.RFQ[0].RFQLine[0].Item.ItemID = new ItemID[1];
            grfq.DataArea.RFQ[0].RFQLine[0].Item.ItemID[0] = new ItemID();
            grfq.DataArea.RFQ[0].RFQLine[0].Item.ItemID[0].ID = new ID();
            grfq.DataArea.RFQ[0].RFQLine[0].Item.ItemID[0].ID.Value =
txt_ProductId.Text;


//logic to receive and display the response
        }
```

Figure 9.3: Code is executed at the supplier graphical user interface to generate the XML document.

Similarly the following code is executed at the module activation controller web service method of the supplier agent.

```
[WebMethod]
    public ShowRFQ ReceiveGetRFQ(GetRFQ getrfq)
    {
        string mfrid, prodid, rfqid, messagerfq, priceupdate;
        DateTime reqdd, EarliestShipDate;
        decimal reqquant, AvailableQuant, sum = 0, linedupprod = 0,
ModifiedunitPrice;
        bool rfq;
        string[] ProdCap, ProdCur, Data;
        int availabledays, actualavailabledays, carrierdays = 3;


        RFQ MyRFQ = new RFQ();
        RFQLine myRFQLine = new RFQLine();
        ItemID myItemId = new ItemID();
        ID myId = new ID();

        System.Collections.ArrayList myArraylist =
getrfq.DataArea.RFQCollection;
        System.Collections.IEnumerator myEnum =
myArraylist.GetEnumerator();

        while (myEnum.MoveNext())
        {
            MyRFQ = (RFQ)myEnum.Current;
        }

        //myEnum = myRFQLine.Item.ItemIDCollection.GetEnumerator();
        myEnum = MyRFQ.RFQLineCollection.GetEnumerator();

        while (myEnum.MoveNext())
        {
            myRFQLine = (RFQLine)myEnum.Current;
        }
        //get the required quantity data
        reqquant = myRFQLine.Quantity.MixedValue;
```

```
        //calculate the required Delivery date
        reqdd = DateTime.Parse(myRFQLine.RequiredDeliveryDateTime);

        myEnum = myRFQLine.Item.ItemIDCollection.GetEnumerator();

        while (myEnum.MoveNext())
        {
            myItemId = (ItemID)myEnum.Current;
        }
        //retrieve the Product Id
        prodid = myItemId.ID.MixedValue;

        //retreive retailer ID.
        myEnum =
myRFQLine.RequesterParty.PartyIDs.IDCollection.GetEnumerator();
        while (myEnum.MoveNext())
        {
            myId = (ID)myEnum.Current;
        }

        mfrid = myId.MixedValue;

      //logic to prepare the response

        //preparing the ShowRFQ

        ShowRFQ srfq = new ShowRFQ();

        RFQ srfqmyRFQ = new RFQ();


        RFQLine srfqmyRFQLine = new RFQLine();

        srfqmyRFQLine.UnitPrice.PerQuantity.MixedValue = ModifiedunitPrice;
        srfqmyRFQLine.RequiredDeliveryDateTime =
EarliestShipDate.ToShortDateString();
```

```
        Description srfqmyDescription = new Description();
        srfqmyDescription.MixedValue = messagerfq;
        srfqmyRFQLine.DescriptionCollection.Add(srfqmyDescription);

        srfqmyRFQ.RFQLineCollection.Add(srfqmyRFQLine);

        srfq.DataArea.RFQCollection.Add(srfqmyRFQ);
        //srfqmyRFQHeader.DescriptionCollection.
        srfq.ApplicationArea.BODID.MixedValue = rfqid;

        return (srfq);
    }
```

Figure 9.4: Code is executed at the supplier Module Activation Controller

The above code demonstrates how the module activation controller web service method "ReceiveGetRFQ" retreives values from the XML message and then how it prepares a new ShowRFQ message.

10. CONCLUSION

This thesis has attempted to advance the art of supply chain management by using agent technology to automate the process of decision making in supply chain. Apractical agent based supply chain architecture was illustrated to show the benefits and feasibility of using agent technology in supply chain management.

In the beginning, MASCOT "Multi Agent Supply Chain Coordination Tool" agent architecture was selected as the base architecture for agent implementation. Its modular internal architecture was the best fit for implementation using web services. Its emphasis on the modular encapsulation of problem solving knowledge within independent knowledge sources and its implementation using web services enables one to add functionality by changing only the code in the knowledge source web service. The module activation controller can then increase the agent functionality by easily orchestrating the construction and revision of solutions by activating this new services from the knowledge source module web service.

This thesis has attempted to advance the art of supply chain management by using agent technology to automate the process of decision making. Apractical agent based supply chain architecture was illustrated to show the benefits and feasibility of using agent technology in supply chain management.

In the beginning, MASCOT "Multi – Agent Supply Chain Co ordination Tool "agent architecture was selected as the base architecture for agent implementation. Its modular internal architecture was the best fit for implementation using web services. Its emphasis on the modular encapsulation of problem solving knowledge within independent knowledge sources and its implementation using web services enables one to add functionality by changing only the code in the knowledge source web service. The module activation controller can then increase the agent functionality by easily orchestrating the construction and revision of solutions by activating these new services from the knowledge source module web service.

Open Applications Group Integration Specifications (OAGIS) based standard business object documents were used to demonstrate communication amongst agents. OAGIS currently has diverse implementations in the field of ecommerce, manufacturing, logistics, CRM, ERP with over 102 OEM implementation and 27755 total dealerships (OAGi). It is a universally adopted standard which reduces the development time and cost. OAGIS 9.1 schemas used in this implementation are free to download and use and they provide 434 different business object documents.

The databases developed in this application used Microsoft SQL Server 2005 Express, which is a free, easy to use, redistributable version of SQL Server 2005 for building simple data driven applications. It can be used directly from inside the Visual Studio 2005 environment, which was used to develop this application. This enabled the use of specialized C# classes for database interaction. It can be easily upgraded to more sophisticated versions of SQL servers which provides scope to extend this work into full, large scale industry application.

Finally, when the implementation was completed, the GUI for each interacting partner in the architecture provided the means to demonstrate the results of using agent technology. It shows how much time is saved and the accuracy with which the decisions can be made. The ability to automatically complete the responses to different request from interacting partners and taking decisions regarding pricing, shop floor production plan, cancellation policies etc saves the organization valuable human hours which can be used in more productive ways for the organization.

In the process of developing this thesis, the use of OAGIS based standard business object documents has shown how XML standards can be used for communication. It also demonstrates the advantages of using the modular structure provided by adopting MASCOT architecture.

One major advantage of this architecture and web service implementation is that different agents developed by other organizations using different technology other then web services,

can easily interact with our agents by simply querying the web service. It does not force the organizations participating in a particular supply chain and using agent technology, to have the same architecture for their agents.

There is tremendous scope for future extension of this work. To start with, one can implement real life interaction amongst the interacting agents. For example, the module activation controller of the manufacturer agent can generate a GetRFQ document and send it directly to the supplier who can then automatically respond by sending back the ShowRFQ document. This work, currently demonstrates this kind of interaction using the GUI for each agent.

Secondly, the blackboard functionality of the MASCOT agent which has not been implemented in this work can be implemented. This will enable sequential queuing selective functioning of various jobs to be performed by the agent. Currently, the agent is only able to process one request at a time. By using the blackboard and asynchronous processing in web services, multiple requests can be stored on the blackboard and then based on pre developed priority plan, the stored tasks can be processed at a later time.

Also, multiple agents can be developed for the same organization. Each agent can be assigned to a particular level within the organization. Based on an agents level, it should perform different tasks. For example, three different levels can be created for agents. At the highest level, agents can be responsible for strategic decision making or high level lateral control. At the second level, the agents can control the low level lateral control and at the lowest level the agents should preside over the shop floor. Establishing communication amongst agents, not only at same levels, but also at different levels can further help to improve supply chain efficiency.

Reference:

OAGi. "Who uses OAGIS?" 2006. Open Applications Group. 2007
http://www.openapplications.org/downloads/Presents/2007%200104%20Who%20Uses%20OAGIS.pdf.

Appendix 1

C#

The programming language used to implement the thesis was C#, which comes along with the commercial package 'Visual Studio 2005' provided by Microsoft. C# is a true object oriented programming language with properties like objects, classes, polymorphism, data encapsulation, Inheritance etc. Modular structures can be created easily and dependencies among the various blocks can be reduced using OOPs (Advantages and Disadvantages of OOP). The code can be maintained and modified easily as per requirements and required further enhancements. Moreover it has a huge set of component libraries in place to serve common processes like xml parsing, web service implementation, SQL database connections etc. The architecture of web services provided by C# is quite developed and the user is only required to implement the functionality of the web service. C# also provides a high level of data security and hence can be used in industry for B2B communications. Used along with the .Net Framework , it can be developed to build a web interface if required. The web application can be built on guidelines of MVC architecture. It has a very good Integrated Development Environment and the ease of creating classes is truly remarkable. Moreover it can be easily configured for the Microsoft SQL database which serves as back end. The web services communicate using the SOAP (Standard Object Access Protocol).

Database functionality using C#

This appendix explains how database functionality was implemented in this thesis using C# classes

The two main classes used in implementing database functionality are 'SqlCommand' and 'SqlSelect'. 'System.Data' assembly and 'SqlClient' namespace is added to the web service 'Service.cs' file. 'System.Data' assembly is used t It is the .Net Framework Data Provider for SQL Server. It describes a collection of classes used to access a SQL Server (System.Data.SqlClient Namespace). This namespace inherits the 'SqlConnection' and 'SqlCommand'. 'SqlConnection' class represents an open connection to the SQL Server database (SqlConnection Class). 'SqlCommand' class represents a Transact-SQL statement or stored procedure to execute against a SQL Server Database (SqlCommand Class).

An object 'con' of the SqlConnection class was created using the following syntax:

```
SqlConnection con = new SqlConnection("Data Source = IMERT215\\SQLEXPRESS; Initial Catalog = Carrier_Vilesh;Integrated Security = True")
```

The parameters passed to the constructor specify the connection string to be used to connect to the SQL Server. Data Source property specifies the location of the SQL Server 2005 Express. Initial Catalog property specifies the specific database to be accessed and Integrated Security property is set to true to establish a secure connection to the database. More properties can be added to the connection string if required.

Similarly, an object of the SqlCommand class is also created everytime an SQL query is to be executed. An example of this is given below:

```
SqlCommand cmd = new SqlCommand("SELECT Distance, TimetoDestinationRoad, TimetoDestinationAir" + " FROM States" + " WHERE State = @state", con)
```

The string value in the above syntax is the SQL query that will be executed. 'con' is the connection string described above. Specific values are then inserted into the 'cmd' object using:

```
cmd.Parameters.AddWithValue("@state", value)
```

Now, two different methods were adopted for execution of SQL queries. One method was to execute "SELECT" queries and second one to execute "UPDATE/INSERT/DELETE" queries.

Method to execute SELECT queries

First the connection string is opened using "con.Open()" method. An object of the SqlDataReader classis then created as shown below

SqlDataReader reader = cmd.ExecuteReader();

The ExecuteReader() method is used execute command that returns rows (SqlCommand Class).

After all the selected values are retrieved using "Object.Add(reader.GetValue())" the connection is closed using "con.Close()".

The following is the code used in the thesis

```
protected string[] SelectMethod(SqlCommand cmd)
  {
    try
     {
       Application.Lock();
       string[] returnValue;
       ///using list to store the information
       List<string> lst = new List<string>();

       using (con)
       {
          try
          {
             ///open the connection
             con.Open();
             ///retrieve data from the datasource
             SqlDataReader reader = cmd.ExecuteReader();
             while (reader.Read())
             {
```

```
            for (int i = 0; i < reader.FieldCount; i++)
            {
                lst.Add(reader.GetValue(i).ToString());
            }
        }
    }
    finally
    {
        ///if no value is returned then add a null value to the list.
        if (lst.Count == 0)
        {
            lst.Add(null);

        }
        ///close the connection
        con.Close();
    }
  }

  returnValue = lst.ToArray();
  return returnValue;
 }
 finally
 {
  Application.UnLock();
 }
}
```

Method to execute UPDATE/INSERT/DELETE queries

The execution of this code is similar to the previous method. First the connection string is opened. Then 'ExecuteNonQuery' method is executed on the object of the 'SqlCommand' class. Finally the connection is closed. Given below is the code for the same.

```
//method to update data in the database

  protected void UpdateInsertDelete(SqlCommand cmd)
  {
    try
    {
      Application.Lock();

      using (con1)
      {
        con1.Open();
        cmd.ExecuteNonQuery();
        con1.Close();
      }
    }
    finally
    {
      Application.UnLock();
    }
  }
```

The 'ExecutenonQuery()' method is used to execute commands such as Transact – SQL INSERT, DELETE, UPDATE and SET statements (SqlCommand Class).

References:

Advantages and Disadvantages of OOP. 2005. 04 September 2007 <http://wiki.tcl.tk/13398>.

SqlCommand Class. 2007. 09 September 2007 <http://msdn2.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.aspx>.

SqlConnection Class. 2007. 09 September 2007 <http://msdn2.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection(VS.71).aspx>

System.Data.SqlClient Namespace. 2007. 08 September 2007 <http://msdn2.microsoft.com/en-us/library/system.data.sqlclient(VS.71).aspx>.

Appendix 2

eXtensible Markup Language

Growing popularity of the internet has caused a marked increase in the need to search, display, manipulate and exchange information. Markup languages attempt to standardize data, following the underlying concept that both the person marking the content and the person processing the content have both agreed on what the markup symbols mean (C. Zender, 2006). It is similar to a teacher with a red pencil marking up a student's paper.

1. History of Markup languages (B. Marchal, 2000):

While writing a word document, word processing requires the user to specify the appearance of the text. For example the user selects a typeface and its boldness. He can also place a piece of text at a given position on the page and more, which is stored as special codes with the text. This method of storing data is called procedural markup. However this approach has a few disadvantages. It does not record the structure of the document, is inflexible and error prone. From this reference point, Markup language evolved into generic coding with the advent of macros. Macros use calls to external formatting procedures. A generic identifier (GI) or tag is attached to each text element and formatting rules are further associated with tags. A formatter then processes the text and produces a document in the format of the output device. Generic coding method provides the advantage of higher portability, more flexibility and is closer to describing the structure of a document.

The next step in the chain of evolution was the standard Generalized Markup Language (SGML) which extended generic coding. Unlike generic coding, SGML markup describes the document's structure, not its appearance. The markup in SGML confirms to a model or schema. SGML does not impose its own tag set, but proposes a language for authors to describe the structure of their documents and mark them accordingly.

Although SGML does not impose a structure on documents, standard committees, industry groups build on SGML and describe standard document structures as SGML applications. Some document structures are maintained as public standards in the form of SGML Document Type Definitions, example HTML. Hyper Text Markup Language, HTML is one set of tags that follow the rules of SGML.

Although HTML is a very powerful tool to display the contents of a text, it still has limitated capacity in terms of describing the structure of the document. eXtensible Markup Language, XML helps us to overcome this difficulty.

2. What is XML???

XML, eXtensible Markup Language, is a data description language, a subset of Standard Generalised Markup Language. It is a format designed to bring structured information to the Web, allowing users to define their own set of markup tags relating to the content of their documents, thus delivering both extensibility and potential for validation (A. Zisman, 2000).

XML is a standard used to create a text-based structure for storing information. A meta language, that is, a programming language used to describe information (S. Taylor, 2003). XML is maintained by the World Wide Web Consortium (W3C) and is a open standard and open source.

XML separates form from content, thus allowing the same information to be viewed in different ways. It describes the structure of the text within a document i.e. it contains explicit rules to determine where specific document structure begins and ends. XML provides a data standard that can encode the content, semantics and schemata for a wide variety of cases – whether as a wire format for sending data between client servers, a transfer format for sharing data between applications, or a persistent storage format on disc. Above all XML seeks to achieve a compromise between flexibility, simplicity and readability by both humans and machines.

3. XML versus HTML:

The main difference between XML and HTML is that HTML was designed to display data and to focus on how data looks, whereas XML focuses on describing the data as it is. An HTML document rendered in a web browser is human readable. XML is aimed towards being both human and machine readable.

How XML syntax differs from HTML:

- New tags may be defined at will
- Tags may be nested to arbitrary depth
- May contain an optional description of its grammar

The information contained hereon is obtained mainly from the following two sources:

XML Tutorial. W3Schools. 2006. April 21, 2006
<http://www.w3schools.com/xml/default.asp>

Benoi^t Marchal. XML By Example. Indiana: Que, 2000.

Any exceptions are specified.

All the examples mentioned in the document are self coined.

4. Getting Started with XML Markup:

Consider the following XML code.

```
1.    <?xml version="1.0" encoding="UTF-8"?>
2.    <!—adopted from Benoi^t Marchal. XML By Example. Indiana: Que, 2000.-->
3.    <address_book>
4.      <entry>
5.        <name>John Doe</name>
6.        <address> complete postal address
7.          <street>197 Kimbal Drive</street>
8.          <region>NY</region>
9.          <postal_code>14623</postal_code>
10.         <locality>New York</locality>
11.         <country />   <!-- default value is "USA"-->
12.       </address>
13.       <tel preferred = "true">585-354-2620</tel>
14.       <tel>585-597-8607</tel>
15.       <email href = "mailto:jdoe@emailaholic.com"/>
16.     </entry>
17.     <entry>
18.       <name><fname>jack</fname><lname>Smith</lname></name>
19.       <tel>585-354-2896</tel>
20.       <email href="mailto:jsmith@emailaholic.com"/>
21.     </entry>
22.   </address_book>
```

Listing 1: XML Document for address book.

The above XML document represents an address book with only two entries. The document consists of character data and markup. Both of which are represented by text. The character data contains the information and the markup records the structure of the document.

The first line (line 1) in the document is the XML declaration - defines the XML version and the character encoding used in the document. The second line (line 2) is a comment. The next line (line 3) describes the root element of the document i.e. address-book. The next 13 lines (line 4 to line 16) describe a single entry in the address book. Lines 5 to 15 describe the children elements of the element <entry> which in turn is the child element of the root element <address-book>. Lines 17 to 21 describe a second entry.

Figure 1 shows the hierarchical tree structure of the address_book XML document from Listing 1. It shows the root element address_book and the corresponding child elements.

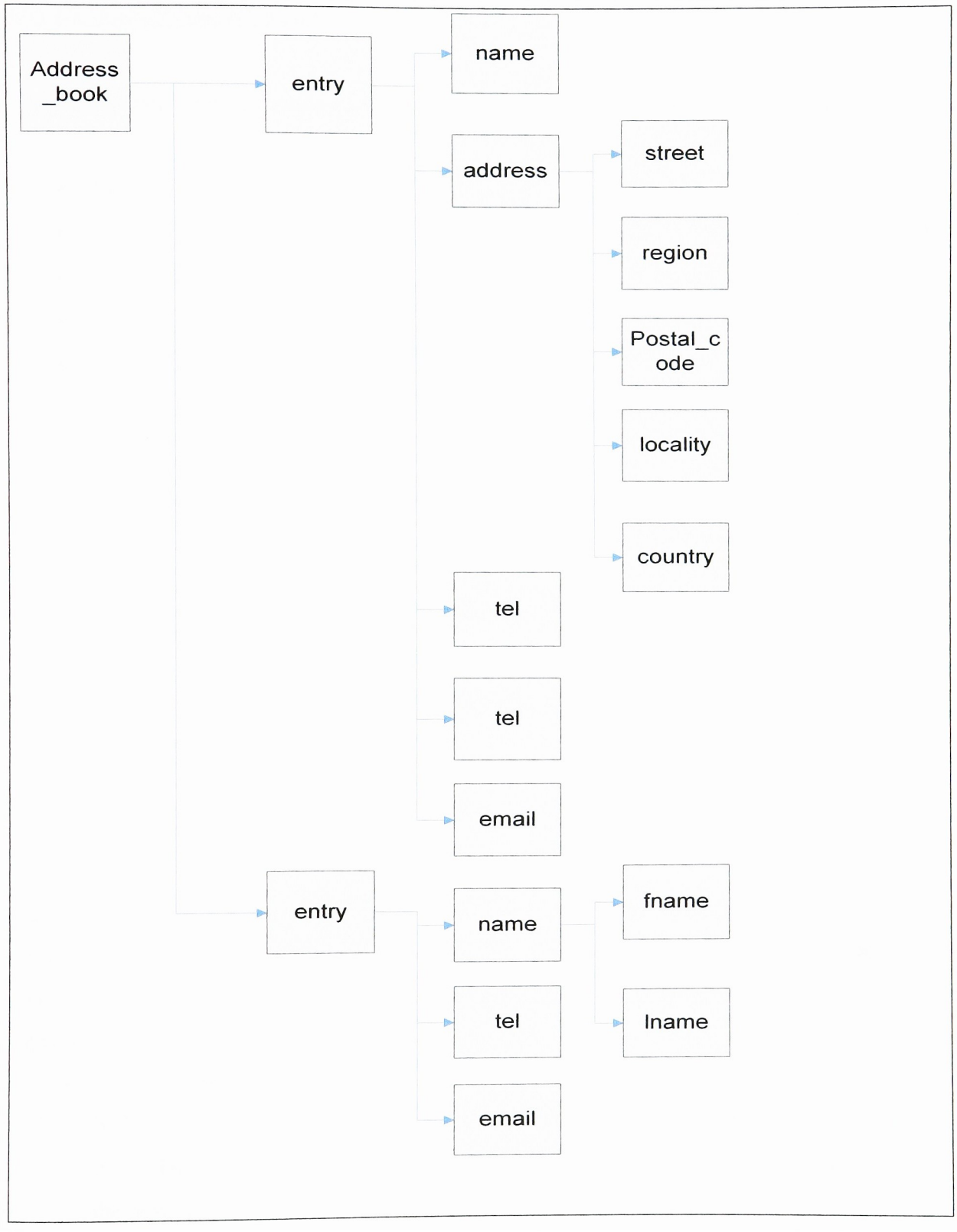


Figure 1: Hierarchical tree structure for the XML document of Listing 1.

4.1 Building Blocks (C. Zender, 2006):

All XML documents are made up by the following simple building blocks:

- Elements
- Tags
- Attributes
- Entities
- PCDATA
- CDATA

The following is a brief explanation of each of the building blocks:

4.1.1 Tags:

Tags are used to markup elements. A starting tag like <element_name> marks up the beginning of an element, and an element tag like </element_name> marks the end of an element. XML tags are case sensitive.

4.1.2 Elements:

Elements are the main building blocks of XML. An XML element is everything from (including) the elements start tag to (including) the elements end tag. All XML elements must have a closing tag. Each element has a name and content. XML elements have the following specific characteristics

- XML Elements are extensible

  XML validating methods (schema, DTD) designed for Listing 1, can validate the document even if new content is added to the listing. The schema will simply ignore the new content.

- Elements have relationships
    - Elements are related as parent and children. In listing 1. <address-book> is the root element. <entry> is the child element of <address_book>. <street>,<region>,<postal_code>,<locality> and <country> are the child elements of <entry> and are sister elements of each other.

- Elements have content
    - Elements can have different content types. An element can have element content, mixed content, simple content, or empty content.
      In Listing 1. <address_book> has element content, because it contains other elements. <address> has mixed content as it has both text and XML elements. <street> has simple content.

    - Elements that have no content are known as empty elements. They are included in the document for the value of their attributes. Lines 15 and 20 are examples of empty elements.

- Element Naming

  XML elements must follow these naming rules:
    - Names can contain letters, numbers, and other characters
    - Names must not start with a number or punctuation character
    - Names must not start with the letters xml (or XML, or Xml, etc)
    - Names cannot contain spaces

Words are not reserved in XML. Any words can be used to make the element names descriptive.

4.1.3 Attributes:

Attributes provide extra information about elements. Attributes are always placed inside the starting tag of an element. Attributes have a name and a value. The names follow the same rules as element names. Elements can have one or more attributes in the start tag, and the name is separated from the value by the equal character. The value of the attribute is enclosed in a double or single quotation marks.

Line 13 in Listing 1, i.e. `<tel preferred = "true">585-354-2620</tel>` shows the element **tel** having a "preferred" attribute of value **true.**

4.1.4 Entities:

Entities are variables used to define common text. Complex documents are often split over several files: the text, the accompanying graphics etc. However, XML organizes documents physically in entities. Entities are inserted in the document through entity references, the name of the entity between an ampersand character and a semicolon. For application, the entity reference is replaced by the content of the entity. Assume an entity "us" which has the value "United States", then the following two lines are equivalent:

<country>&us;</country>

<country>United States</country>

The following entities are predefined in XML:

Table 1: Predefined Entities in XML

Entity References	Character
<	<
>	>
&	&
"	"
&apos	'

4.1.5 PCDATA:

PCDATA means parsed character data. Character data is the text found between the start tag and the end tag of an element. PCDATA is the text that will be parsed by the parser (A PARSER is a piece of software which can take any Document Type Definition/XML Schema and generate from it a software capable of validating any document invoking that DTD). Tags inside the text are treated as markup and entities are expanded.

4.1.6 CDATA:

CDATA means character data. CDATA is text that is not parsed by a parser. Tags inside the text are not treated as markup and entities are not expanded.

5. XML Validation:

Any XML document confirming to the XML syntax rules is termed as a "well formed" document. Well formed documents have the right mix of start tags and end tags, attributes are properly quoted, entities are acceptable, character sets are properly used. Valid XML documents are stricter in that they not only follow the syntax rules, but also comply with a specific structure. This compliance is achieved via either Document Type Definition (DTD) or XML Schema.

5.1 Document Type Definition:

A document type definition (DTD) defines legal building blocks of an XML document. The basic building blocks of a DTD are similar to the XML documents, however the syntax for a DTD is different from the syntax for XML documents

Listing 2 is the Document Type Definition for the address book XML Document in Listing 1. It confirms to the element structure followed in listing 1, which can be easily verified through figure 1.

Listing 2 is the DTD for the address book in listing 1:

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <!--top level element, the address book is a list of enteries -->
3.  <!ELEMENT address_book (entry)+ >
4.  <!-- an entry is a name followed by addresses, phone numbers, etc. -->
5.  <!ELEMENT entry (name,address*,tel,fax*,email*)>
6.  <!-- name is made of string,first name and last name. This is a very flexible model to
    accomodate exotic name -->
7.  <!ELEMENT name (#PCDATA | fname | lname)* >
8.  <!ELEMENT fname (#PCDATA) >
9.  <!ELEMENT lname(#PCDATA) >
10. <!-- definition of the address structure if several addresses, the preferred attribute
signals the "default" one -->
11. <!ELEMENT address (street, region?,postal_code, locality, country) >
12. <!ATTLIST address preferred (true | false) "false" >
13. <!ELEMENT street (#PCDATA) >
14. <!ELEMENT region(#PCDATA) >
15. <!ELEMENT postal_code(#PCDATA) >
16. <!ELEMENT locality (#PCDATA) >
17. <!ELEMENT country (#PCDATA)>
18. <!-- default values cannot be specified for elements in a DTD, but they can be defined in
    a schema-- >
19. <!ATTLIST country std CDATA "USA">
20. <!-- phone, fax and email, same preferred attribute as address -->
21. <!ELEMENT tel (#PCDATA) >
22. <!ATTLIST tel preferred (true | false) "false" >
23. <!ELEMENT fax (#PCDATA) >
24. <!ATTLIST fax preferred (true | false) "false" >
25. <!ELEMENT email empty >
26. <!ATTLIST email href CDATA #REQUIRED >
```

Listing 2: DTD for the address book

5.1.1: Declaring Elements:

The following table summarizes the DTD language Elements.

Table 2: Syntax to Declare XML building block "Element" in a DTD (XML_Tutorial. W3Schools. 2006. April 21, 2006 <http://www.w3schools.com/xml/default.asp>)

Heading/ Subheading	Description	Syntax	Example
Declaring of Elements	XML Elements are declared with an Element Declaration	<!ELEMENT element_name category> or <!ELEMENT element_name (element_content)>	<!ELEMENT address_book (entry)+ > or <!ELEMENT fname (#PCDATA) >
Empty elements	Declared with the keyword empty	<!ELEMENT element_name EMPTY>	<!ELEMENT email empty >
Elements with only chatachter data.	Declared with #PCDATA in parenthesis	<!ELEMENT element_name (#PCDATA)>	<!ELEMENT fname (#PCDATA) >
Element with ANY content	Declared with keyword ANY, can contain any combination of parsable data.	<!ELEMENT element_name ANY>	

Element with Child Element	Elements with one or more children are defined with the name of the children elements inside parenthesis.	<!ELEMENT element_name (child_element_name, child_element_nam,...>	<!ELEMENT entry (name,address*,tel,fax*,email*)> Note: the child elements must appear in the XML document in the same sequence in which they are declared in the DTD.
Declaring only one occurrence of the same element		<!ELEMENT element_name (child_name)>	<!ELEMENT entry (name,address*,tel,fax*,email*)> Child element note must occur only once in the element entry
Declaring minimum one occurrence of the same element	+ sign used along with the child elements name	<!ELEMENT element _name (child_name+)	<!ELEMENT address_book (entry)+>
Declaring zero or more occurrence of the same element	* sign used along with the name of the child element	<!ELEMENT element_name (child_elemen*)	<!ELEMENT entry (name,address*,tel,fax*,email*)
Declaring zero or one occurrence of the same element	? sign used along with the name of the child element	<!ELEMENT element_name (child_element?)	
Declaring either/or content	Used to specify certain specific content	<!ELELMENT element_name (content 1| content 2)>	
Declaring mixed content	Element can contain more then one	<!ELEMENT element_name (data	<!ELEMENT name

	type of data.	type|data type …)*>	(#PCDATA|fname|lname)> Element name can contain either parsable character data or can contain child element fname or lname

5.1.2 Declaring Attributes:

Attributes are used to provide extra information about the elements. Their declaration is done using the word "ATTLIST", as can be seen in the following syntax declaration:

```
<!ATTLIST element_ name  attribute_name attribute_type default value>
```

Line 12 in Listing 2, <!ATTLIST address preferred (true|false) "false"> declares an attribute "preferred" for the element "address". The attribute type declared is enumeration. There are many different attribute types available for DTD, a list of which is entailed in Table 3.

Table 3: Different Attribute types

Value	
CDATA	The value is character data
(en1|en2|..)	The value must be one from an enumerated list
ID	The value is a unique id
IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation
xml:	The value is a predefined xml value

The default type can have the following values:

Table 4: Default Attribute types

Value	Explanation
Value	The default value of the attribute
#REQUIRED	The attribute value must be included in the element
#IMPLIED	The attribute does not have to be included
#FIXED value	The attribute value is fixed

5.1.3 DTD - ENTITIES

Entities are variables used to define shortcuts to common text. Entity references are references to entities. Entities can be declared internal or external.

5.1.3.1 Internal Entity Declaration:

Syntax:

```
<!ENTITY entity-name "entity-value">
```

DTD Example:

```
<!ENTITY friend "John Doe.">
```

XML example:

```
<name>&friend </friend>
```

5.1.3.2. External Entity Declaration:

Syntax:

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

5.2 XML Schema:

The Document Type Definition language utilized by the XML community is derived from SGML. It allows for constraints to be placed on the types of data that can be contained in a XML document (B. Sutor, 2000). However, since DTD is derived from the SGML camp, which was focused on the printed page, DTD's are of limited use when describing structured data that is to be passed between computer systems. While DTD allows for the specification of broad types of data contained in an XML document, it does not allow for the definition and validation of complex data types. XML schema overcomes these disadvantages. (A,M.Futrell, 2001)

The key advantage of XML schema over DTD's is that XML schemas are written in XML, DTD's on the other hand are not written in XML. Table 5 lists some key comparisons between XML Schema and DTDs:

Table 5: Some Differences Between XML Schema and DTD (A.M.Futrell, 2001)

XML Schema	DTD (Document Type Definition)
Utilizes XML Syntax	Has a Syntax that is different from XML
Defines several basic data types that can be refined to define new types	Has no well known basic elements
Allows the definition of new types	No way to define new data types
Can define many a broad cardinality range utilizing minOccurs and maxOccurs	1,zero or more(*), and 1 or more (+) are the only possible cardinalities for an element.
Can define very detailed types such as an integer between 0 and 10	Allows only basic element definition with no way to restrict values
Cab scope the definition of elements within other elements	All elements are of global scope

Relationship between an XML Schema built with XML Schema and an XML Document that uses that schema is similar to the relationship between a class and an object in a object oriented language (Y. Shohoud, 2000). A schema defines a class of documents (those documents which conform to the schema). An XML document that uses a schema is a instance document of that schema (A.M.Futrell, 2001). The XML Schema language is also referred to as XML Schema Definition (XSD).

Listing 3 is the XML Schema for the address book XML Document in Listing 1. It confirms to the element structure followed in listing 1. "address_book" is the root element. "entry" is the child element of "address_book". This is illustrated in the hierarchical tree structure depicted in figure 1.

Listing 3 is the XML Schema for the address book in Listing 1:

```
1. <?ml version="1.0" encoding="ISO-8859-1"?>
2. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3.    <xs:element name = "address_book">
4.       <xs:complextype>
5.        <xs:sequence>
6.          <xs:element name = "entry" maxOccurs = "unbounded">
7.            <xs:complexType>
8.             <xs:sequence>
9.               <xs:element name= "name" type = "xs:string">
10.                 <xs:sequence>
11.                  <xs:element name="fname" minoccurence = "0"/>
12.                  <xs:element name="lname" minoccurence = "0"/>
13.                 </xs:sequence>
14.              </xs:element>
15.              <xs:element name= "address" mixed = "true">
16.                <xs:complexType>
17.                  <xs:sequence>
```

```
18.                     <xs:element name ="street" type = "xs:string"/>
19.                     <xs:element name ="region" type = "xs:string"/>
20.                     <xs:element name ="postal_code" type ="xs:integer"/>
21.                     <xs:element name ="locality" type ="xs:string"/>
22.                     <xs:element name =" country" type ="xs:string" default =
                        "USA"/>
23.                   </xs:sequence>
24.                 </xs:complexType>
25.               </xs:element>
26.                <xs:element name ="tel" maxOccurs ="4">
27.                   <xs:complexType>
28.                     <xs:attribute name="preferred" default ="false">
29.                       <xs:restriction base = "xs:string">
30.                         <xs:enumeration value = "true/false"/>
31.                       </xs:restriction>
32.                     </xs:attribute>
33.                   </xs:complexType>
34.                 </xs:element>
35.               <xs:element name="email" type="eemail"/>
36.                 <xs:complexType name="eemail">
37.                   <xs:attribute name="href" type ="xs:string"/>
38.                 </xs:complexType>
39.               </xs:sequence>
40.               </xs:complexType>
41.             </xs:element>
42.         </xs:sequence>
43.     </xs:complextype>
44.   </xs:element>
45. </xs:schema>
```

Listing 3: XML Schema for Address Book.

Every XML schema has a root element "<schema>"

```
<?xml version="1.0"?>
<xs:schema>
...
...
</xs:schema>
```

This schema element may contain some attributes, as seen below:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
...
...
</xs:schema>
```

Line xmlns:xs = http://www.w3.org/2001/XMLSchema specifies that the data types and elements used in the schema come from the http://www.w3.org/2001/XMLSchema namespace. the elements and datatypes coming from this namespace should also have xs as its prefix.

Line targetNamespace = http://www.w3schools.com specifies that the elements defined in this schema come from the http://www.w3schools.com smaespace.

The fragment xmlns = http://www.w3schools.com specifies this as the default namespace.

The fragment elementFormDefault = "qualified" indicates that element used by the XML instance document should be namespace qualified.

Section 5.2.1 to section 5.2.6 explain the syntax to describe the different XML document elements in an XML Schema.

5.2.1 XSD SIMPLE ELEMENTS:

A simple element in a XSD is similar to its counterpart in an XML document, i.e. it can contain only text. It cannot contain any elements or attributes. However, the text content can be of different types, all of which are included in the XML schema definition (Boolean, string, date etc). Custom type data can also be defined, which lets us put restrictions (facets) on the content of the data.

Syntax for defining:

```
<xs: element name ="xxx" type = "yyy"/>
```

Where xxx is the name of the element and yyy is its type.

Example:

```
<street> 197 Kimbal Drive </street>
<postal_code> 14623</ postal_code >
```

Equivalent Schema declaration:

```
<xs: element name = "street" type = "xs:string"/>
<xs: element name = "postal_code" type = "xs:integer"/>
```

In the above element definitions xs:string, xs:integer are examples of built in data types. Some common examples are

xs:string

xs:decimal

xs:integer

xs:Boolean

xs:date

xs:time

It is possible to define a default value or a fixed value set for simple elements. A default value is automatically assigned to the element, when no other value is specified.

```
<xs:element name = "country" type = "xs:string" default = "USA"/>
```

A fixed value is automatically assigned to an element. One cannot specify any other value.

5.2.2 XSD ATTRIBUTE:

Syntax of attribute:

```
<xs:attribute name = "xxx" type = "yyy"/>
```

where xxx is the name and yyy is the type of the attribute.

```
<xs:attribute name ="preferred" default ="false">
```
(line 28. listing 3)

Default and fixed values can also be defined for attributes in a manner similar to the element definition. In addition to the default and fixed values, attributes can also be optional or required. By default all attributes are optional. However they can be defined using the keyword "use" as follows

```
<xs:attribute name = "lang" type = "xs:string" use = "optional"/>
```

Attributes can also be made required as follows

```
<xs: attribute name = "lang" type = "xs:string" use = "required"/>
```

5.2.3 XSD RESTRICTIONS/FACETS:

XML allows for constraints to be placed on the types of data that can be used in a document. Restrictions or facets are a mechanism to define these constraints. Restrictions are used to control acceptable elements or attributes.

The following table gives a description of the different restrictions that can be placed on datatypes.

Table 6: Restrictions on Datatypes.

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than

	or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

Following example shows the use of the enumeration restriction on the attribute "preferred" for the element "tel".

```
<xs:element name ="tel" maxOccurs ="4">
 <xs:complexType>
  <xs:attribute name="preferred" default ="false">
   <xs:restriction base = "xs:string">
    <xs:enumeration value = "true/false"/>
   </xs:restriction>
  </xs:attribute>
 </xs:complexType>
</xs:element>
```

Listing 4: Enumeration Restriction example

5.2.4 XSD Complex Elements:

A complex element besides containing text data, also contains other elements and attributes. In XML schemas, complex type elements can be based on existing complex type elements. For example, line 15 to 25 can be re written as follows

```
<xs:element name ="address"  type="fullpersoninfo"/>
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs: element name="street" type ="xs:string"/>
    <xs:element name="region" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name ="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base ="personinfo">
      <xs:sequence>
        <xs:element name ="postal_code" type ="xs:integer"/>
        <xs:element name ="locality" type ="xs:string"/>
        <xs:element name ="country" type ="xs:string" default = "USA"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Listing 5: Complex type Elements based on existing Complex type Elements.

Complex Type "personinfo" is first defined and then used in the complex Type "fullpersoninfo" using the keyword "extension base".

There are four kinds of complex elements:

1) empty element
2) elements that contain only other elements
3) elements that contain only text
4) elements that contain both other elements and text.

5.2.4.1 Empty Element:

An element which does not have any content between the opening and closing tags is an empty element. Empty elements can contain attributes.

Line 15 in Listing 1, <email href = mailto:jdoe@emailaholic.com/>, is an empty element and line 35 to 38 in Listing 3 are its equivalent XML schema validation.

5.2.4.2 XSD Complex Type: Element Only

An "elements only" complex type contains an element that contains only other elements.

Element "entry" in listing 1 is an element only, complex element.

5.2.4.3 XSD Complex Type: Text – Only Elements:

This type of elements can contain both text and attributes. Line 13 in listing 1 is an example. Its equivalent schema definition is on lines 26-34 in listing 3.

5.2.4.4 XSD Complex Type: Mixed Content:

A mixed complex type contains attributes, elements and text. Lines 6 to 12 in listing 1 are example of Mixed content.

5.2.5 XSD Complex Type Indicators:

Indicators are used to control how elements are used in XML documents. There are three major types of indicators as follows:

1) Order Indicators : used to define how elements should occur

    a. All : specifies that child element can occur in any order and that each child element should occur only once.

    b. Choice : specifies that either one child element can occur or another.

    c. Sequence: specifies that child elements must occur in a specific sequence

2) Occurrence Indicators: used to define how frequently an element can occur

    a. maxOccurs: specifies the maximum no. of times an element can occur.

    b. minOccurs: specifies the minimum no. of time an element can occur.

3) Group Indicators: used to define related sets of elements.

    a. Element groups:

        Element groups are defined with group declaration as below:

```
<xs:group name="persongroup">
  <xs:sequence>
        <xs:element name ="street" type = "xs:string"/>
        <xs:element name ="region" type = "xs:string"/>
        <xs:element name ="postal_code" type ="xs:integer"/>
  </xs:sequence>
</xs:group>
```

The above example defines a group named "persongroup", that defines a group of elements that must occur in an exact sequence

b. Attribute groups:

Attribute groups are defined with attribute Group declaration

5.2.6 XSD The <any> Element:

The <any> element allows the extension of XML document with elements not specified by the schema.

5.2.7 XSD the <anyAttribute> Element:

The <anyAttribute> element allows the extension of XML document with attributes not specified by the schema.

6. XML NAMESPACES (Benoi^t Marchal. XML By Example. Indiana: Que, 2000)**:**

XML is extensible. In a distributed environment, improper management of this extensibility causes conflicts. XML namespaces provide a mechanism to avoid element name conflict by placing the names of the elements in a more global context.

Since XML names are not predefined in XML, a name conflict occurs when two documents use the same element names. Consider the following XML documents, listings 6 and 7 which describe business partners of a company.

```
<business_partner>
        <information>
                <name>XYZ corporation</name>
                <address> Rochester, NY </address>
                <type> Manufacturer </type>
        </information>
</business_partner>
```

Listing 6: Type of Business partner

```
<business_partner>
        <information>
                <product>talcum powder</product>
                <price>$5</price>
        </information>
<business_partner>
```

Listing 7: Type of the Business partners product.

Listing 6 provides information about the type of business partner i.e. manufacturer, whereas listing 7 provides information about the product which a business partner makes. If these two XML documents were combined (listing 8), there would be an element name conflict because

both documents contain element "business_partner" with different content and definition. XML namespaces helps to solve this problem.

```
<business_partner>
        <information>
                <name>XYZ corporation</name>
                <address> Rochester, NY </address>
                <type> Manufacturer </type>
        </information>
        <information>
                <product>talcum powder</product>
                <price>$5</price>
        </information>
</business_partner>
```

Listing 8: Combined Listing of Listing 6 & 7.

The XML namespace (xmlns) Attribute:

The XML namespace attribute is placed in the start tag of an element and has the following syntax

```
xmlns:namespace_prefix= "namespaceURI"
```

When a namespace is defined in the start tag of an element, all child elements with the same namespace belong to the same namespace. It is important to note that, the namespace is not used by the parser to lookup information. The only purpose is to give the namespace a unique name.

Default Namespace:

Defining a default namespace for an element does prevent defining a namespace for every child element. It has the following syntax.

```
xmlns: "namespaceURI"
```

Listing 9 shows the combined data listing of Listing 6 & 7 with Namespaces, which helps to identify the difference between the information tags.

```
<business_partner xmlns: cinfo= "http://www.yahoo.com"
                    xmlns: pinfo = "http://www.hotmail.com">
      <cinfo: information>
            <name>XYZ corporation</name>
            <address> Rochester, NY </address>
            <type> Manufacturer </type>
      </information>
      <pinfo: information>
            <product>talcum powder</product>
            <price>$5</price>
      </information>
</business_partner>
```

Listing 9: Combined listing of Listing 6 & 7 with Namespaces

Reference:

Anthony M. Futrell Jr. "The W3Cs XML Schema", North Carolina State University, 2001. April 21, 2006 <http://home.ec.rr.com/ncbeach/ncsu/csc513/xmlschema.html>

A. Zisman. "An Overview of XML", Computing and Control Engineering Journal, August 2000. April 21, 2006 < http://ieeexplore.ieee.org/iel5/2218/18764/00866909.pdf?arnumber=866909>

B. Sutor. "Making XML Ready for E-Business", XML Journal, Volume. 1 issue 3, June 15 2000. pp. 22. April 21, 2006 <http://xml.sys-con.com/read/40055.htm>

Cynthia, Zender. "Markup 101: Markup Basics", Paper TU 12. Electronic. SAS Institute, Inc., Cary, NC. April 21, 2006 < http://whitepapers.zdnet.co.uk/0,39025945,60157537p-39000960q,00.htm>

Stephanie Taylor. "A Quick Guide to... XML", Interlending and Document Supply, Volume 31. Number 3, 2003. pp 187-191, April 21, 2006 <http://www.emeraldinsight.com/Insight/ViewContentServlet?Filename=Published/EmeraldFullTextArticle/Pdf/1220310306.pdf >

XML Tutorial. W3Schools. 2006. April 21, 2006 <http://www.w3schools.com/xml/default.asp>

Benoi^t Marchal. XML By Example. Indiana: Que, 2000.

Y. Shohoud. "XML's Grand Schema", XML Magazine", Summer 2000. May 15, 2006 <http://www.xmlmag.com/upload/free/features/xml/2000/03sum00/ys0300/ys0300.asp>

Appendix 3

OAGIS

In today's business, organizations buy packaged software's for their key business functions. This results in more options and more functionality for the organizations. However, to leverage full capabilities of these applications, they need to be integrated together (Connelly). This chapter describes the horizontal message based architecture presented by the Open Applications Group, Inc. It also explains the details of the structure of Business Object Documents. The content in this document is from (BUSINESS OBJECT DOCUMENT MESSAGE ARCHITECTURE)

1. Introduction:

The Open Applications Group, Inc. (OAGi) is a non-profit industry consortium comprised of many of the most prominent stakeholders in the business software component industry throughout the world. It was formed in February of 1995 as a nonprofit consortium of leading enterprise application software developers. The mission of the Open Applications Group is to promote the easy and cost-effective integration of business application software components for enterprises (Connelly).

In order to achieve this interoperability between disparate systems, disparate companies and disparate supply chains, there must be a common message architecture that provides a common understanding for all (BUSINESS OBJECT DOCUMENT MESSAGE ARCHITECTURE). Once a messaging architecture has been agreed upon, different messages can be sequenced together to form scenarios which provide detail step by step exchange of information needed to perform specific tasks. The Open Applications Group Integration Specification (OAGIS) provides these example scenarios that can be used to identify different messages needed to achieve ones needs. OAGIS provides a common message architecture, Business Object document (BOD). BODs are the business messages or business documents that are exchanged between software applications or components; between companies; across supply chains; and between supply chains.

The BOD message architecture is independent of the communication. It can be used with simple transport protocols such as HTTP and SMTP but it can also be used with more

complex transport protocols such as SOAP, ebXML Transport and Routing, or any other Enterprise Application Integration System.

A BOD graphically is as shown in fig 1.

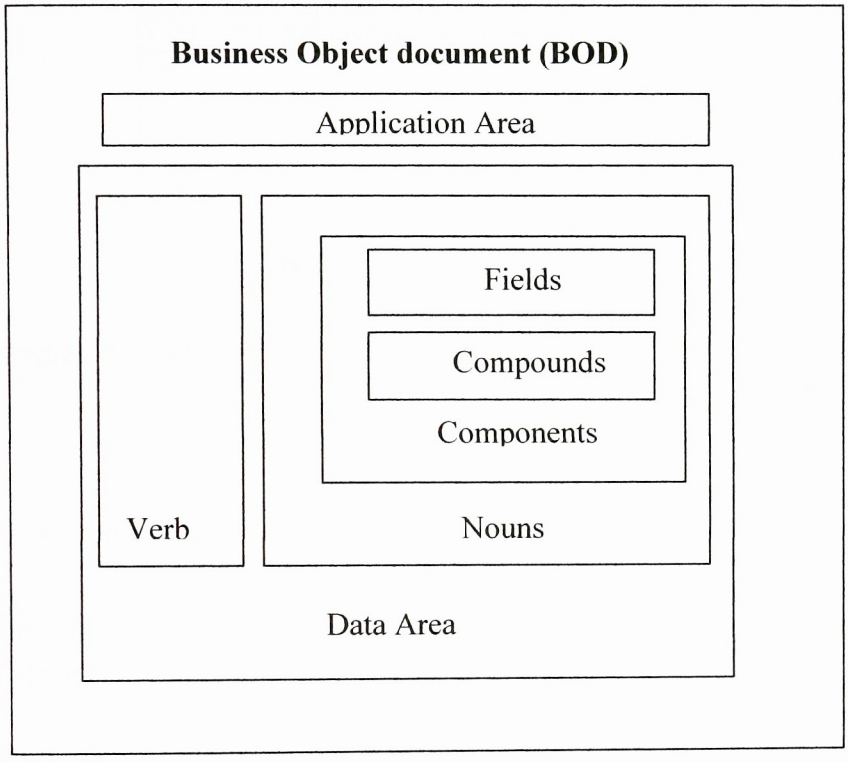


Figure 1: Graphical representation of Business Object Document

These areas are explained in the following sections

2. Business Object Document

General structure for all of Business Object Documents is shown in the following figure 2.

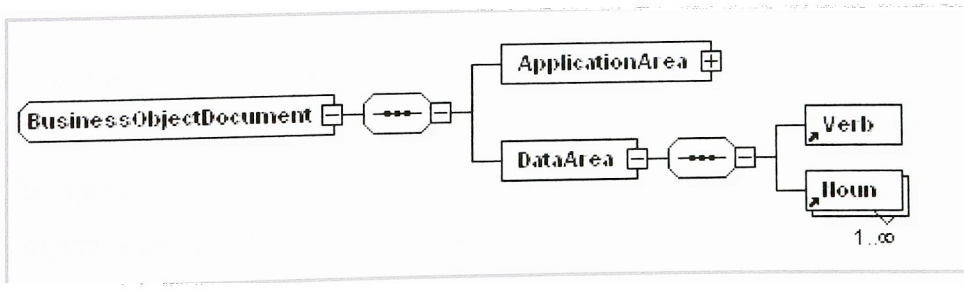


Figure 2: General Structure of Business Object Document

For a given Business Object Document, the generic names (BusinessObjectDocument, Verb, Noun) are replaced by specific names (ProcessPurchaseOrder, Process, and PurchaseOrder)

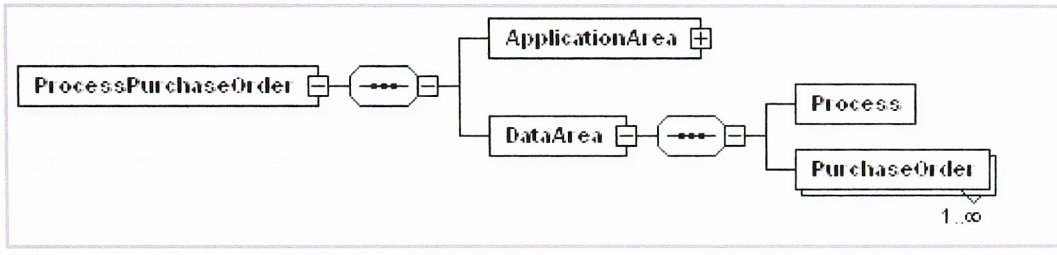


Figure 3: Specific Structure of a Business Object Document

Business Object Document comprises of the following child elements:

- Application Area
- Data Area

The Application Area and Data Area separate the application-specific information common to all BODs from the information that is specific to each BOD.

In addition to these child elements each BOD contains four attributes, the BODs releaseID, versioned, systemEnvironmentCode, and language code.

Release ID

Release ID is used to identify the release of OAGIS that the BOD belongs. For the BODs from OAGIS 9.0 the value of this attribute will be "9.0". The release ID is a required attribute of the BOD.

Version ID

Version ID is used to identify the version of the Business Object Document. Each BOD has its own revision number to specifically identify the level of that BOD, not just the release ID of OAGIS. The specific BOD version number is documented in each chapter of OAGIS. The version ID attribute is an optional attribute of the BOD.

System Environment Code

The System Environment Code is used to identify whether this particular BOD is being sent as a result of a test or as production level integration. Often times as new systems are brought online testing must be performed in a production environment in order to ensure integration with existing systems. This attribute allows the integrator to flag these test messages as such. The environment attribute is an optional attribute of the BOD.

Language Code

The languageCode attributes indicates the language of the data being carried in the BOD message

2.1 Application Area

The Application Area carries information that an application may need to know in order to communicate in an integration of two or more business applications. The Application Area is used at the applications layer of communication. While the integration frameworks Web Services and middleware provide the communication layer that OAGIS operates on top of. As shown in figure 4 each BOD contains one ApplicationArea.

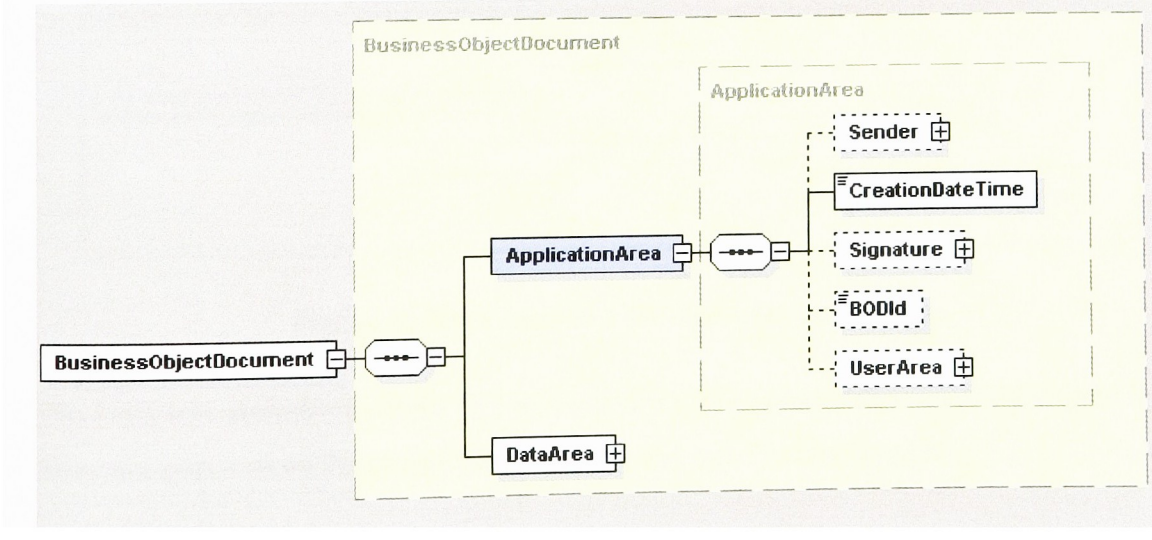


Figure 4: Application Area of a Business Object Document

The ApplicationArea is comprised of following elements:

- Sender
- CreationDateTime
- Signature
- BODId
- UserArea

ApplicationArea serves four main purposes:

1. To identify the sender of the message.
2. To identify when document was created.
3. To provide authentication of the sender through digital signature, if applicable.
4. To uniquely identify a BOD instance. The BODId field is the Globally Unique Identifier for the BOD instance.

2.2Data Area

The Data Area (DataArea) of the Business Object Document contains the instance(s) of data values for the business transaction. For example, to send a Purchase Order or Orders to a business partner; the Data Area will contain Verb (the action) and the Noun (the object) on which the action is to be performed.

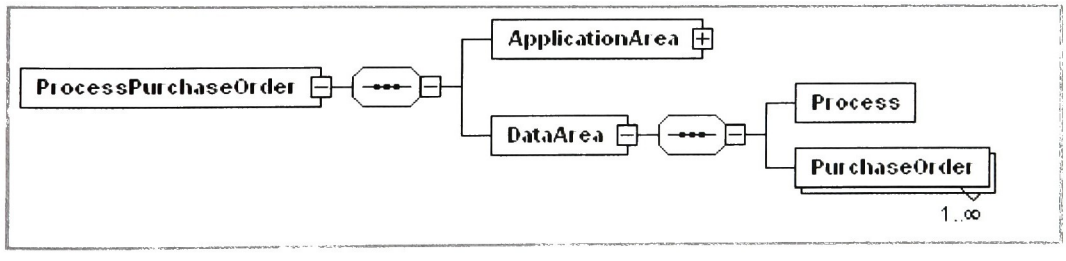


Figure 5: Data Area of a Business Object Document

The DataArea contains a single verb and one or more instances of a noun. As seen in figure 5, there are one or more PurchaseOrder elements and one Process element.

2.2.1 Verb

The verb is the action to be applied to the object (the Noun). Examples of verb include Cancel, Add, Process and Synchronize. Any industrial information that is exclusively related to the action is also stored in the verb.

2.2.2 Noun

Noun is the object or document that is being acted upon. Examples include PurchaseOrder, RequestForQuote, and Invoice. There are different types of verbs or actions that can be performed on a noun. A base noun (e.g PurchaseOrder) contains all the information that might be present on a PurchaseOrder. Nouns are extensible within OAGIS. Additional content, fields, compounds and components can be added to an existing noun.

2.2.3 Components

Components are the large-grained building blocks of a Noun. Components are extensible within OAGIS. Components consist of other Components, Compounds and Fields. Examples of Components are PurchaseOrder Header, Party and Address.

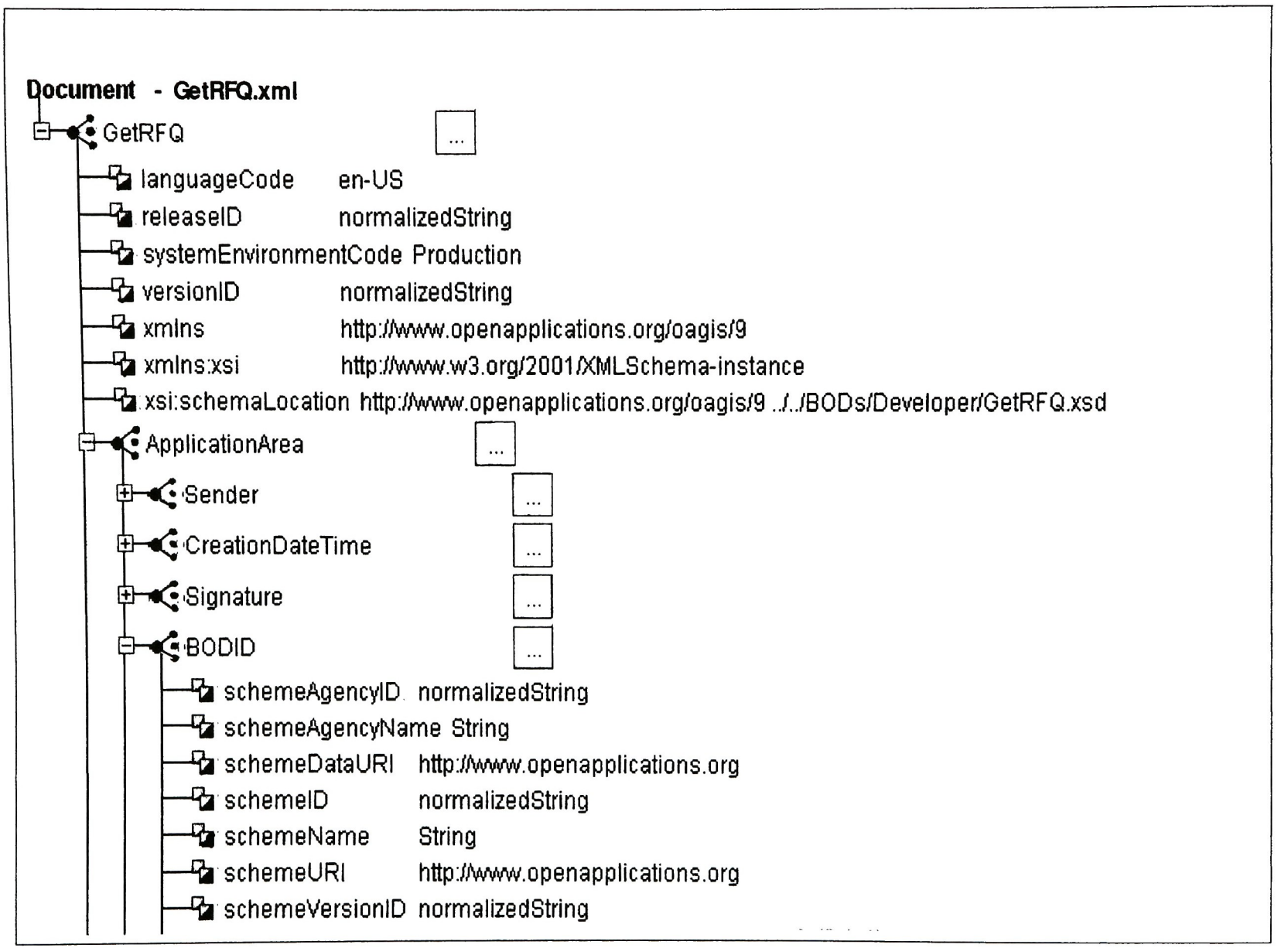
2.2.4 Compounds

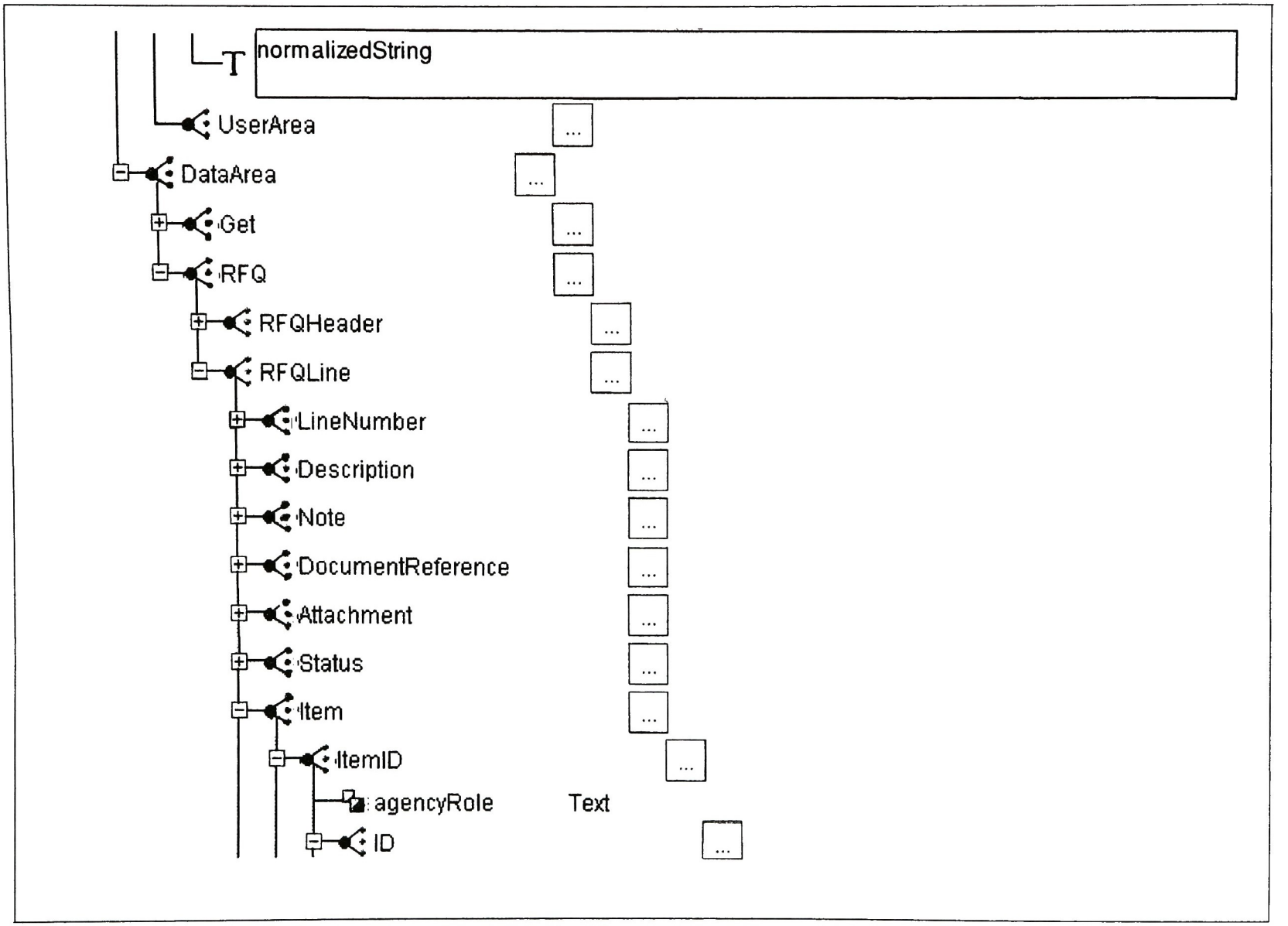
Compounds are a logical grouping of fields (low level elements) that are used across all BODs. They cannot be extended to include new data. Examples of Compounds are Amount, Quantity, DateTime and Temperature.

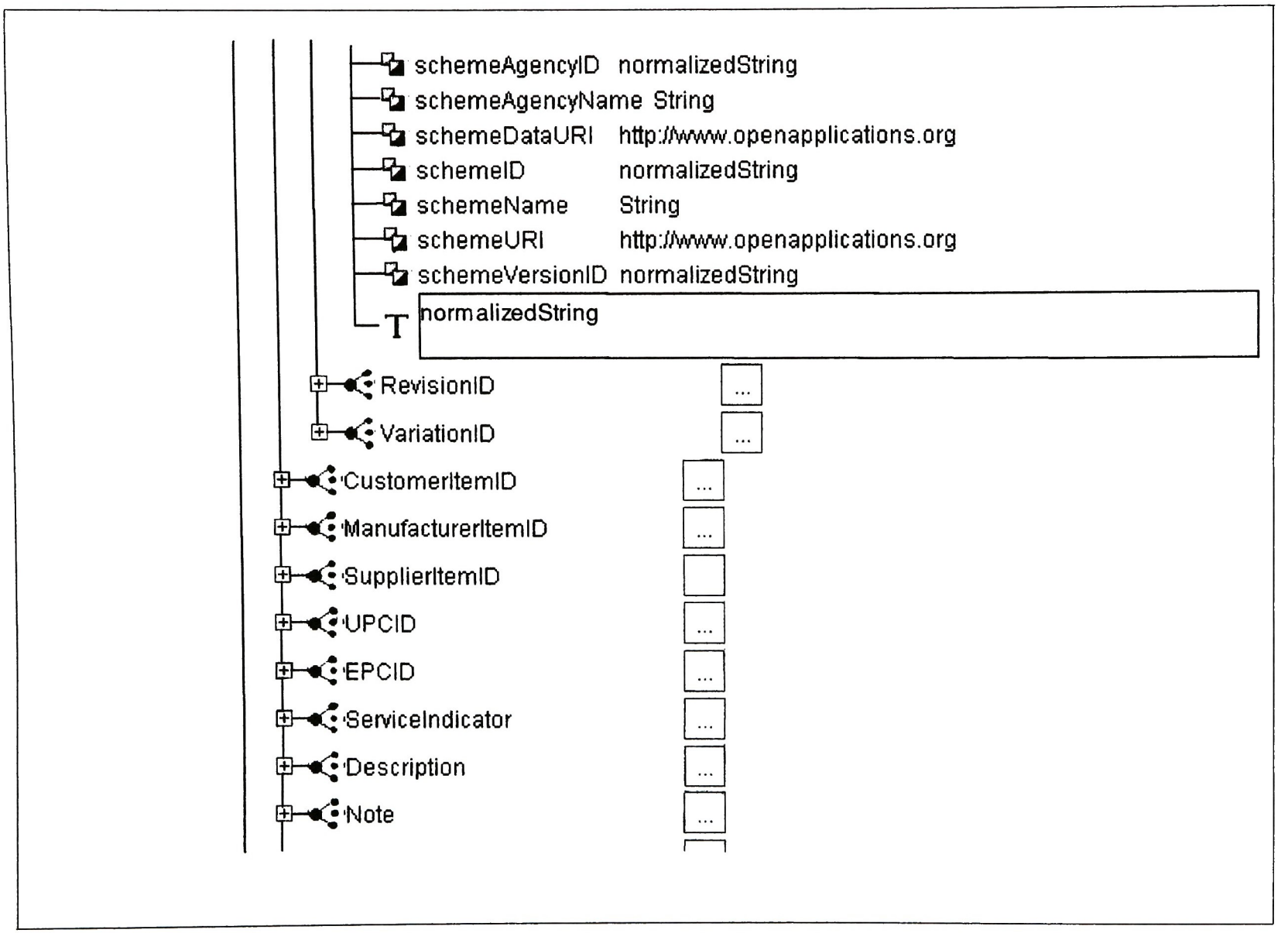
2.2.5 Fields

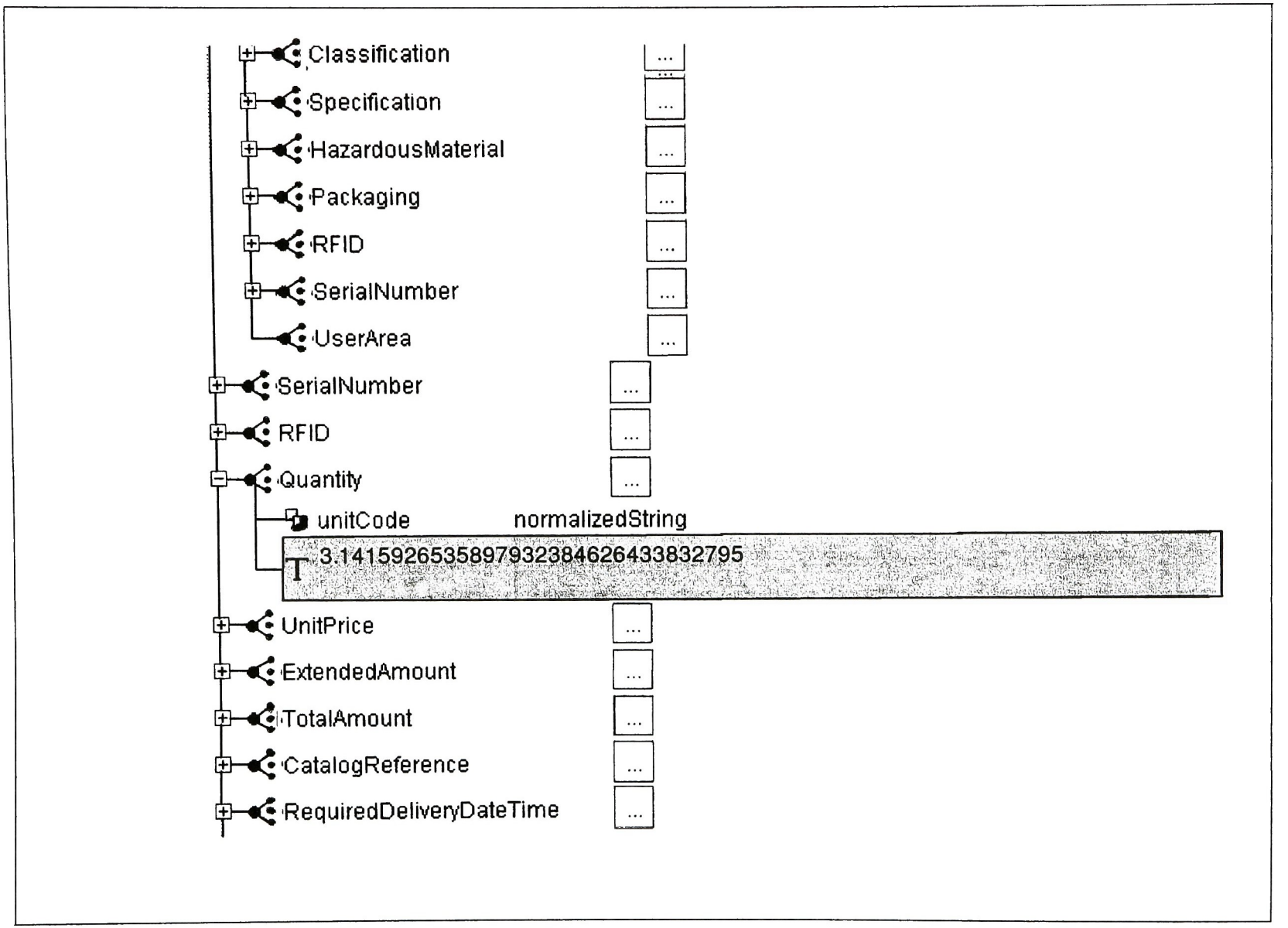
Fields are the lowest level elements used in OAGIS Components and Compounds. These fields can be based on either OAGIS based type or user defined type.

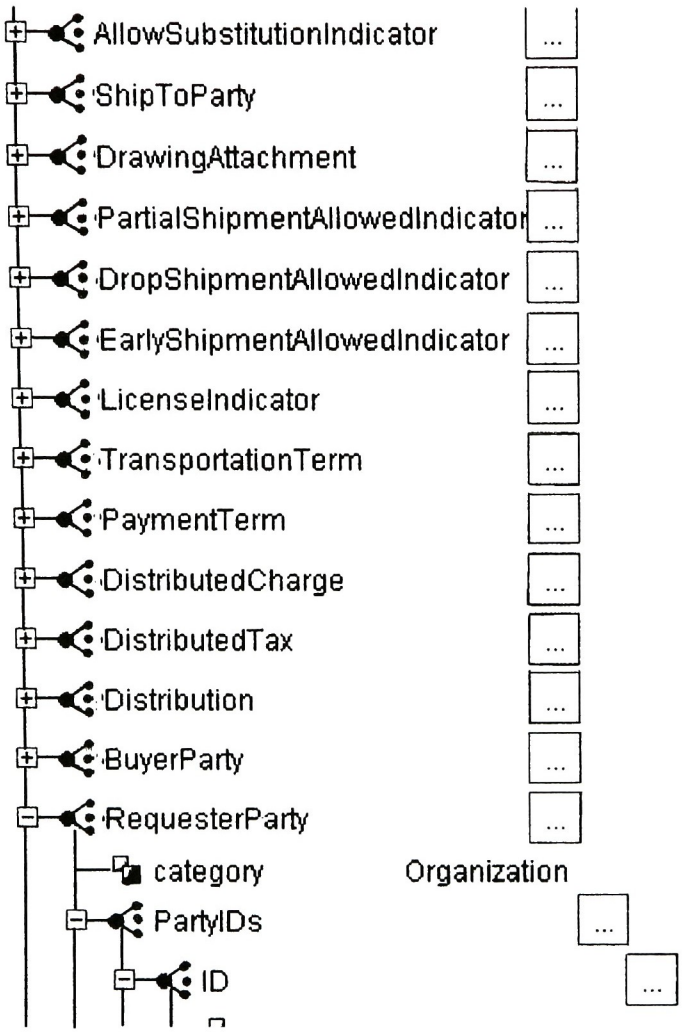
An example of GetRFQ BusinessObject Document is as given below:

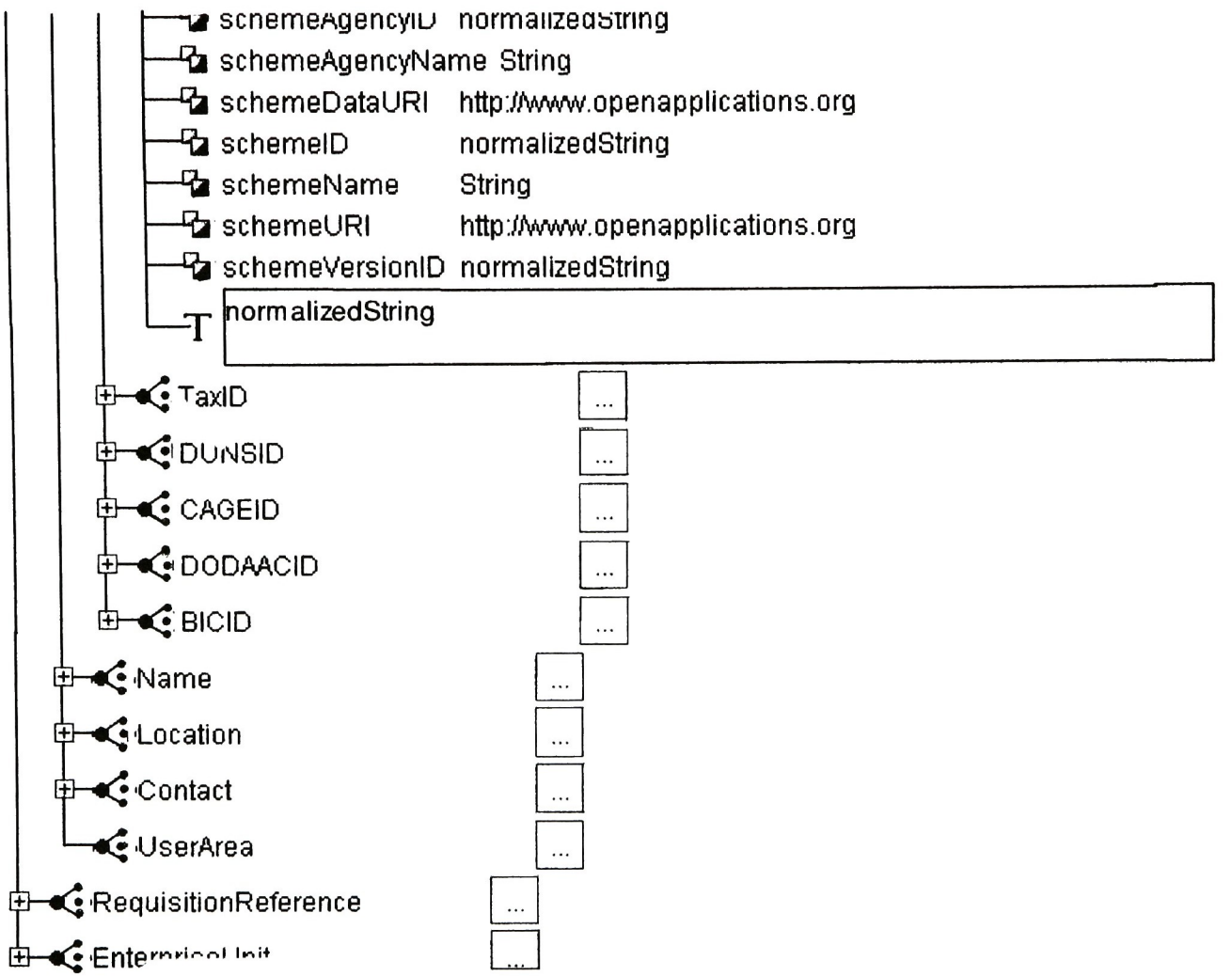

Document - GetRFQ.xml
GetRFQ
languageCode en-US
releaseID normalizedString
systemEnvironmentCode Production
versionID normalizedString
xmlns http://www.openapplications.org/oagis/9
xmlns:xsi http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation http://www.openapplications.org/oagis/9 ../../BODs/Developer/GetRFQ.xsd
ApplicationArea
Sender
CreationDateTime
Signature
BODID
schemeAgencyID normalizedString
schemeAgencyName String
schemeDataURI http://www.openapplications.org
schemeID normalizedString
schemeName String
schemeURI http://www.openapplications.org
schemeVersionID normalizedString

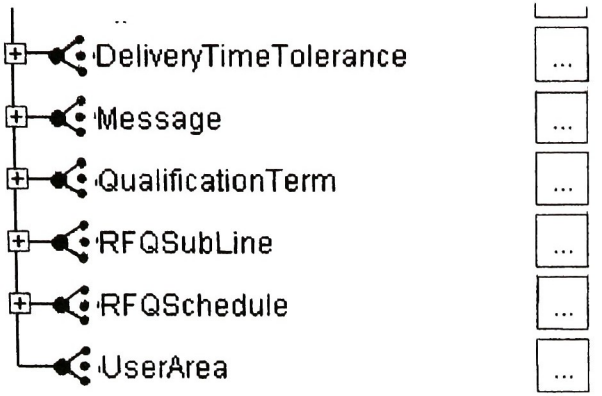

normalizedString
UserArea
DataArea
Get
RFQ
RFQHeader
RFQLine
LineNumber
Description
Note
DocumentReference
Attachment
Status
Item
ItemID
agencyRole
Text
ID


schemeAgencyID normalizedString
schemeAgencyName String
schemeDataURI http://www.openapplications.org
schemeID normalizedString
schemeName String
schemeURI http://www.openapplications.org
schemeVersionID normalizedString
T
normalizedString
RevisionID
VariationID
CustomerItemID
ManufacturerItemID
SupplierItemID
UPCID
EPCID
ServiceIndicator
Description
Note


Classification
Specification
HazardousMaterial
Packaging
RFID
SerialNumber
UserArea
SerialNumber
RFID
Quantity
unitCode
normalizedString
3.14159265358979323846264338327 95
UnitPrice
ExtendedAmount
TotalAmount
CatalogReference
RequiredDeliveryDateTime


AllowSubstitutionIndicator
ShipToParty
DrawingAttachment
PartialShipmentAllowedIndicator
DropShipmentAllowedIndicator
EarlyShipmentAllowedIndicator
LicenseIndicator
TransportationTerm
PaymentTerm
DistributedCharge
DistributedTax
Distribution
BuyerParty
RequesterParty
category
Organization
PartyIDs
ID


schemeAgencyID normalizedString
schemeAgencyName String
schemeDataURI http://www.openapplications.org
schemeID normalizedString
schemeName String
schemeURI http://www.openapplications.org
schemeVersionID normalizedString
normalizedString
TaxID
DUNSID
CAGEID
DODAACID
BICID
Name
Location
Contact
UserArea
RequisitionReference


DeliveryTimeTolerance
Message
QualificationTerm
RFQSubLine
RFQSchedule
UserArea
...
...
...
...
...
...

References:

"BUSINESS OBJECT DOCUMENT MESSAGE ARCHITECTURE." www.openapplicationsgroup.org. 03 September 2007

Connelly, David. "OAGi White Paper.pdf." 2005. www.openapplicationsgroup.org. 03 09 2007 <http://www.openapplications.org/downloads/whitepapers/whitepaperdocs/OAGi%20White%20Paper.pdf>.

Appendix 4

SQL Database Development

This appendix presents the relationship diagrams for manufacturer, supplier and carrier agent databases. It shows all the tables and their fields. An excel table based representation is given which shows all the fields, their data-type, length of the field, are null values allowed for the field.

MANUFACTURER DATABASE:

Figure 1 shown below presents a E-R Diagram for manufacturer agent database, also table 1 details all the relationship defined for the manufacturer database.

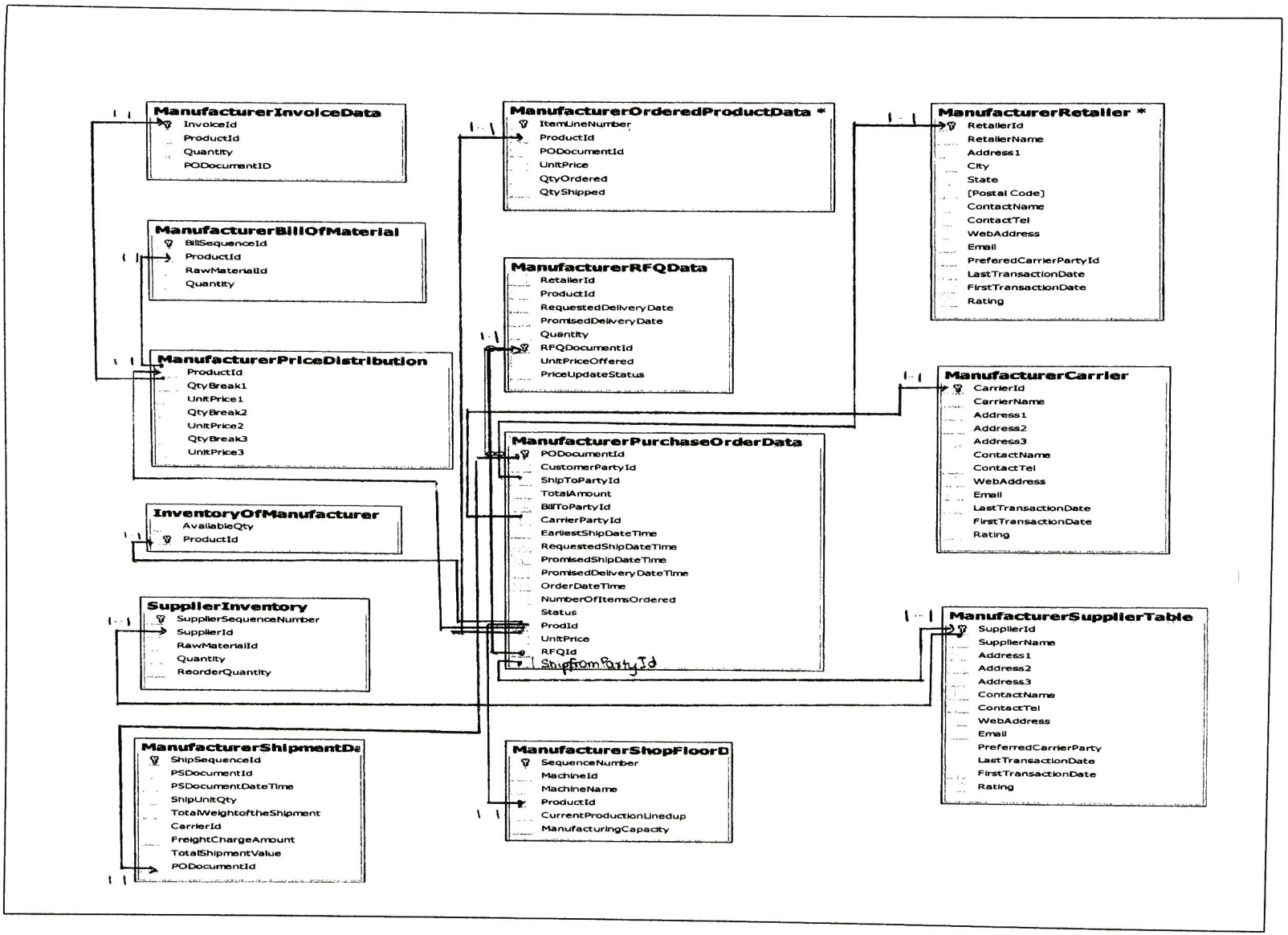


Figure 1: E-R Diagram for Manufacturer

Table 1: Manufacturer Database Relationship Table

Sr. No	Table Name	Field Name	Key Type	Table Name	Field Name	Key Type	Relationship
1.	ManufacturerPurchase-OrderData	PODocumentId	Primary	ManufacturerShip-mentData	PODocumentId	Secondary	One-One
2.	ManufacturerPurchase-OrderData	ShipToPartyId	Secondary	ManufacturerRet-ailer	RetailerId	Primary	One-One
3.	ManufacturerPurchase-OrderData	CarrierPartyId	Secondary	ManufacturerCar-rier	CarrierId	Primary	One-One
4.	ManufacturerPurchase-OrderData	ProdId	Secondary	InventoryOfManu-facturer	ProductId	Primary	One-One
5.	ManufacturerPurchase-OrderData	ProdId	Secondary	ManufacturerPric-eDistribution	ProductId	Primary	One-One
6	ManufacturerPurchase-OrderData	ProdId	Secondary	ManufacturerOrd-eredProductData	ProductId	Secondary	One-One
7.	ManufacturerPurchase-OrderData	ProdId	Secondary	ManufacturerSho-pFloorData	ProductId	Secondary	One-One
8	ManufacturerPurchase-OrderData	RFQId	Secondary	ManufacturerRFQ-Data	RFQDocumentId	Primary	One-One
Sr. No	Table Name	Field Name	Key Type	Table Name	Field Name	Key Type	Relationship
9.	ManufacturerPurchase-OrderData	ShipFromParty-Id	Secondary	ManufacturerSup-plierTable	SupplierId	Primary	One-One
10.	ManufacturerSupplier-Table	SupplierId	Primary	SupplierInventory	SupplierId	Secondary	One-One

11.	ManufacturerPriceDi st-ribution	ProductId	Secondary	ManufacturerBill - OfMaterial	ProductId	Secondary	One-One
12.	ManufacturerPriceDi st-ribution	ProductId	Secondary	ManufacturerInv - oiceData	ProductId	Secondary	One-One

Sr. No.	TableName	FieldName	Allow Nulls?	DataType
1	**ManufacturerPurchaseOrderData**			
		PODocumentId	NO	nchar(20)
		CustomerPartyId	NO	nchar(20)
		ShipToPartyId	YES	nchar(20)
		TotalAmount	YES	money
		BillToPartyId	YES	nchar(20)
		CarrierPartyPreferredId	YES	nchar(20)
		EarliestShipDateTime	YES	datetime
		RequestedShipDateTime	YES	datetime
		PromisedShipDateTime	YES	datetime
		PromisedDeliveryDateTime	YES	datetime
		OrderDateTime	YES	datetime
		NumberOfItemsOrdered	NO	int
		Status	YES	char(20)
		ProdId	NO	nchar(20)
		UnitPrice	NO	money
		RFQId	NO	nchar(25)
2	**ManufacturerSupplierTable**			
		SupplierId	NO	nchar(20)
		SupplierName	NO	char(35)
		Address1	NO	char(100)
		Address2	YES	char(35)
		Address3	YES	char(35)
		ContactName	NO	char(35)
		ContactTel	NO	char(35)
		WebAddress	YES	char(35)
		Email	NO	char(35)
		PreferredCarrierParty	YES	DateTime
		LastTransactionDate	YES	DateTime
		FirstTransactionDate	YES	Integer
		Rating	NO	

3	**ManufacturerCarrier**	**CarrierId**	NO	nchar(18)
		CarrierName	NO	char(35)
		Address1	NO	Char(100)
		Address2	YES	char(35)
		Address3	YES	char(35)
		ContactName	NO	char(35)
		ContactTel	NO	char(35)
		WebAddress	YES	char(35)
		Email	NO	char(35)
		LastTransactionDate	YES	datetime
		FirstTransactionDate	YES	datetime
		Rating	NO	char(10)
4	**ManufacturerShipmentData**			
		ShipSequenceId	NO	int
		PSDocumentId	NO	char(35)
		PSDocumentDateTime	NO	datetime
		ShipUnitQty	NO	int
		TotalWeightoftheShipment	YES	decimal(18, 2)
		CarrierId	NO	nchar(18)
		FreightChargeAmount	YES	money
		TotalShipmentValue	YES	money
		PODocumentId	NO	nchar(20)
5	**ManufacturerPriceDistribution**			
		ProductId	NO	nchar(20)
		QtyBreak1	NO	decimal(18, 2)
		UnitPrice1	NO	money
		QtyBreak2	YES	decimal(18, 2)
		UnitPrice2	YES	money
		QtyBreak3	YES	decimal(18, 2)
		UnitPrice3	YES	money
6	**SupplierInventory**			
		SupplierSequenceNumber	NO	int

		SupplierId	NO	nchar(20)
		RawMaterialId	NO	int
		Quantity	NO	int
		ReorderQuantity	NO	int
7	**ManufacturerOrderedProductData**			
		ItemLineNumber	NO	decimal(18, 2)
		ProductId	NO	nchar(20)
		PODocumentId	NO	nchar(20)
		UnitPrice	NO	money
		QtyOrdered	NO	decimal(18, 2)
		QtyShipped	NO	decimal(18, 2)
8	**ManufacturerShopFloorData**			
		SequenceNumber	NO	int
		MachineId	NO	nchar(20)
		MachineName	YES	char(30)
		ProductId	NO	nchar(20)
		CurrentProductionLinedup	NO	decimal(18, 2)
		ManufacturingCapacity	NO	decimal(18, 2)
9	**InventoryOfManufacturer**	AvailableQty	NO	int
		ProductId	NO	nchar(20)
10	**ManufacturerRFQData**			
		RetailerId	NO	nchar(25)
		ProductId	NO	nchar(20)
		RequestedDeliveryDate	NO	datetime
		PromisedDeliveryDate	YES	datetime
		Quantity	NO	decimal(18, 2)
		RFQDocumentId	NO	nchar(25)
		UnitPriceOffered	YES	decimal(18, 2)
		PriceUpdateStatus	NO	nchar(25)
11	**ManufacturerBillOfMaterial**	BillSequenceId	NO	int
		ProductId	NO	int
		RawMaterialId	NO	int

		Quantity	NO	int
12	**ManufacturerInvoiceData**			
		InvoiceId	NO	int
		ProductId	NO	int
		Quantity	NO	int
		PODocumentID	NO	int
13	**ManufacturerRetailer**			
		RetailerId	NO	nchar(20)
		RetailerName	NO	char(35)
		Address1	NO	char(35)
		City	NO	char(35)
		State	NO	char(35)
		[Postal Code]	NO	char(35)
		ContactName	NO	char(35)
		ContactTel	NO	char(35)
		WebAddress	YES	char(35)
		Email	NO	char(35)
		PreferedCarrierPartyId	YES	int
		LastTransactionDate	YES	datetime
		FirstTransactionDate	YES	datetime
		Rating	NO	char(10)

SUPPLIER DATABASE

Figure 2 shown below presents a E-R Diagram for supplier agent database, also table 2 details all the relationship defined for the supplier database.

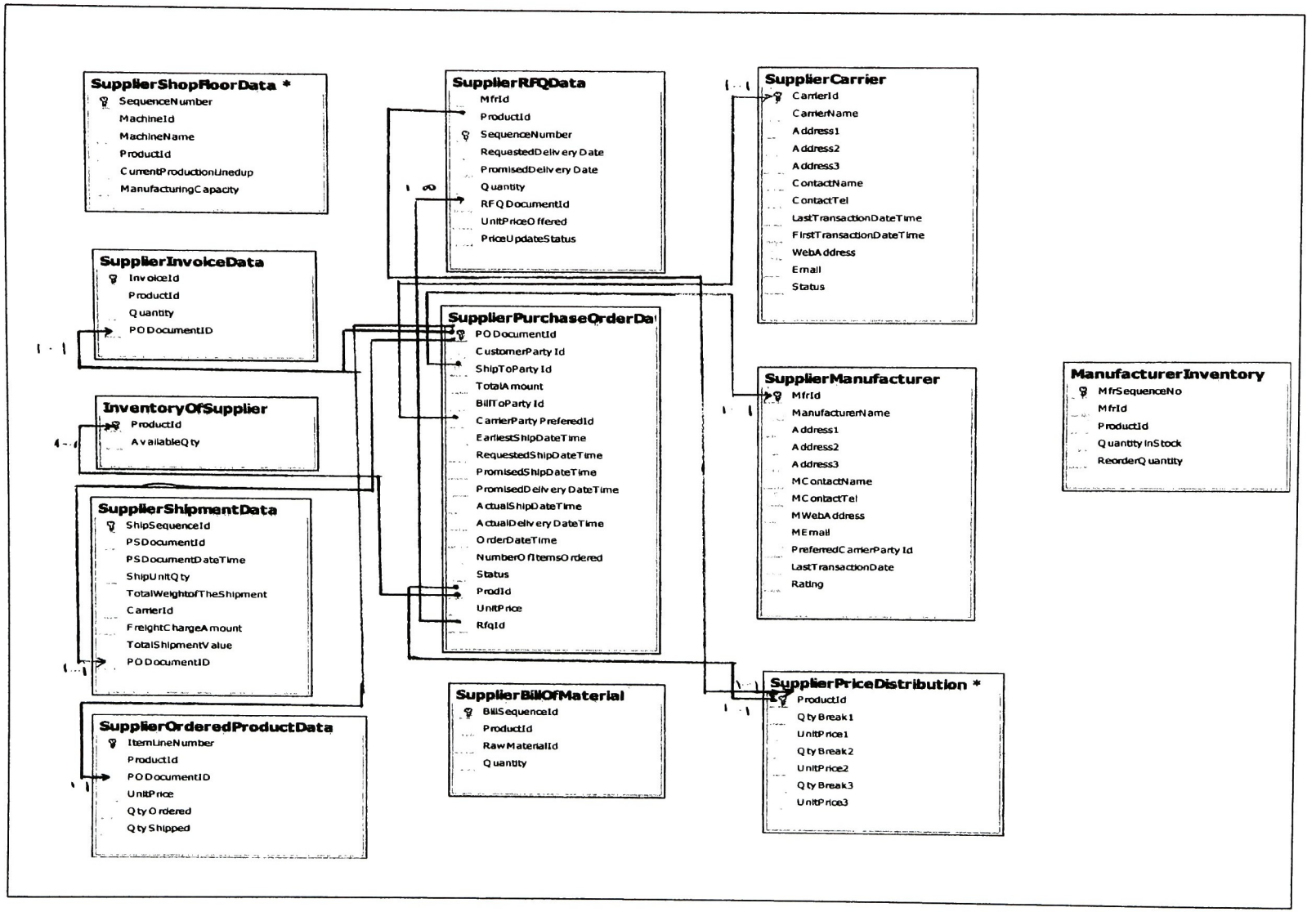


Figure 2: E-R Diagram for Supplier

Table 2: Supplier Database Relationship Table.

Sr. No	Table Name	Field Name	Key Type	Table Name	Field Name	Key Type	Relation-ship
1.	SupplierPurchaseOrder-Data	RfqId	Secondary	SupplierRFQData	RFQDocumentId	Secondary	One – Many
2.	SupplierPurchaseOrder-Data	ShipToPartyId	Secondary	SupplierManufac-turer	MfrId	Primary	One-One
3.	SupplierPurchaseOrder-Data	CarrierPartyPr-eferedId	Secondary	SupplierCarrier	CarrierId	Primary	One-One
4.	SupplierPurchaseOrder-Data	ProdId	Secondary	SupplierPriceDistr-ibution	ProductId	Primary	One-One
5.	SupplierRFQData	ProductId	Secondary	SupplierPriceDistr-ibution	ProductId	Primary	One-One
6.	SupplierPurchaseOrder-Data	ProductId	Secondary	SupplierShopFloo-rData	ProductId	Secondary	One-One
7.	SupplierPurchaseOrder-Data	PODocumentId	Primary	SupplierInvoice D-ata	PODocumentId	Secondary	One-One
8.	SupplierPurchaseOrder-Data	ProdId	Secondary	InventoryOfSupp-lier	ProductId	Primary	One-One
9.	SupplierPurchaseOrder-Data	PODocumentId	Primary	SupplierShipment-Data	PODocumentID	Secondary	One-One
10.	SupplierPurchaseOrder-Data	PODocumentId	Primary	SupplierOrdered P-roductData	PODocumentID	Secondary	One-One

Sr. No.	TableName	FieldName	Allow Nulls?	DataType
1	**SupplierPurchaseOrderData**			
		PODocumentId	NO	nchar(20)
		CustomerPartyId	NO	nchar(20)
		ShipToPartyId	YES	nchar(20)
		TotalAmount	YES	money
		BillToPartyId	YES	nchar(20)
		CarrierPartyPreferedId	YES	nchar(20)
		EarliestShipDateTime	YES	datetime
		RequestedShipDateTime	YES	datetime
		PromisedShipDateTime	YES	datetime
		PromisedDeliveryDateTime	YES	datetime
		ActualShipDateTime	YES	datetime
		ActualDeliveryDateTime	YES	datetime
		OrderDateTime	YES	datetime
		NumberOfItemsOrdered	NO	int
		Status	YES	char(20)
		ProdId	NO	nchar(20)
		UnitPrice	NO	money
		RfqId	NO	nchar(15)
2	**SupplierManufacturer**			
		MfrID	NO	nchar(18)
		ManufacturerName	YES	char(20)
		Address1	NO	Char(100)
		Address2	YES	char(35)
		Address3	YES	char(35)
		MContactName	NO	char(35)
		MContactTel	NO	char(35)
		MWebAddress	YES	char(35)
		MEmail	NO	char(35)
		PreferredCarrierPartyId	YES	nchar(10)
		LastTransactionDate	YES	datetime

		Rating	NO	char(10)
3	**SupplierCarrier**	**CarrierId**	NO	nchar(18)
		CarrierName	NO	Char(50)
		Address1	NO	Char(100)
		Address2	YES	char(35)
		Address3	YES	char(35)
		ContactName	NO	char(35)
		ContactTel	NO	char(15)
		LastTransactionDateTime	YES	datetime
		FirstTransactionDateTime	YES	datetime
		WebAddress	YES	char(35)
		Email	YES	char(35)
		Status	NO	char(15)
4	**InventoryOfSupplier**			
		ProductId	NO	nchar(20)
		AvailableQty	NO	decimal(18, 0)
5	**SupplierShipmentData**			
		ShipSequenceId	NO	decimal(18, 0)
		PSDocumentId	NO	decimal(18, 0)
		PSDocumentDateTime	NO	datetime
		ShipUnitQty	NO	int
		TotalWeightofTheShipment	YES	decimal(18, 0)
		CarrierId	NO	nchar(18)
		FreightChargeAmount	YES	money
		TotalShipmentValue	YES	money
		PODocumentID	NO	nchar(10)
6	**SupplierPriceDistribution**			
		ProductId	NO	nchar(20)
		QtyBreak1	NO	decimal(18, 0)

		UnitPrice1	NO	money
		QtyBreak2	YES	decimal(18, 0)
		UnitPrice2	YES	money
		QtyBreak3	YES	decimal(18, 0)
		UnitPrice3	YES	money
7	**SupplierBillOfMaterial**			
		BillSequenceId	NO	int
		ProductId	NO	nchar(18)
		RawMaterialId	NO	nchar(18)
		Quantity	NO	decimal(18, 0)
8	**SupplierOrderedProductData**			
		ItemLineNumber	NO	decimal(18, 0)
		ProductId	NO	nchar(20)
		PODocumentID	NO	nchar(20)
		UnitPrice	NO	money
		QtyOrdered	NO	decimal(18, 0)
		QtyShipped	NO	decimal(18, 0)
9	**SupplierShopFloorData**			
		SequenceNumber	NO	int
		MachineId	NO	nchar(10)
		MachineName	YES	char(20)
		ProductId	NO	nchar(10)
		CurrentProductionLinedup	NO	decimal(18, 0)
		ManufacturingCapacity	NO	decimal(18, 0)
10	**SupplierInvoiceData**			
		InvoiceId	NO	decimal(18, 0)
		ProductId	NO	decimal(18, 0)
		Quantity	NO	decimal(18, 0)
		PODocumentID	NO	nchar(20)
11	***ManufacturerInventory***	***MfrSequenceNo***	NO	*int*
		MfrId	NO	*nchar(18)*
		ProductId	NO	*nchar(18)*
		QuantityinStock	NO	*decimal(18, 0)*

		ReorderQuantity	NO	*decimal(18, 0)*
12	SupplierRFQData			
		MfrId	NO	nchar(18)
		ProductId	NO	nchar(18)
		SequenceNumber	NO	int
		RequestedDeliveryDate	NO	datetime
		PromisedDeliveryDate	YES	datetime
		Quantity	NO	decimal(18, 2)
		RFQDocumentId	NO	nchar(18)
		UnitPriceOffered	YES	decimal(18, 2)
		PriceUpdateStatus	NO	nchar(18)

CARRIER DATABASE

Figure 3 shown below presents an E-R Diagram for carrier agent database

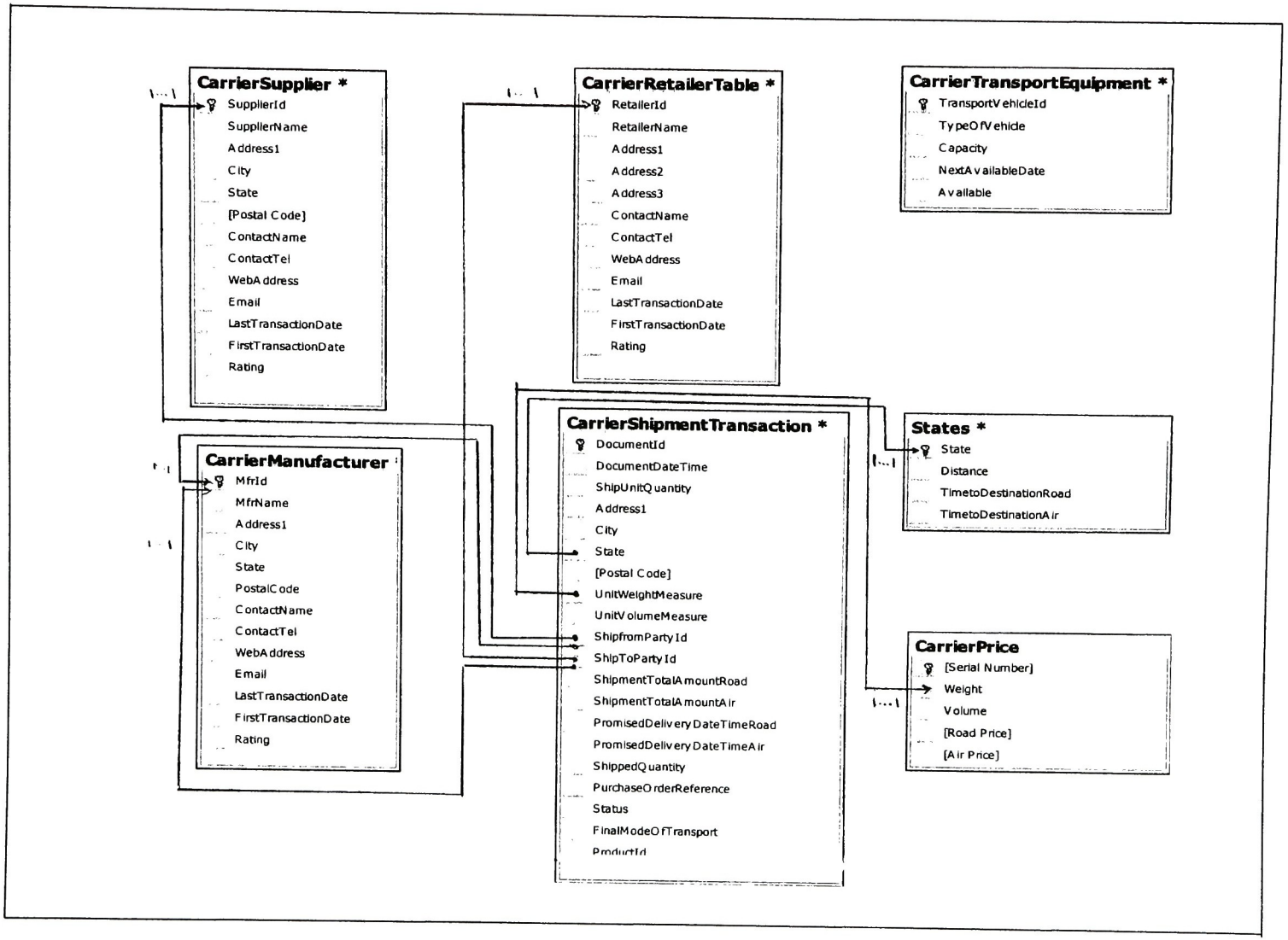


Figure 3: E-R Diagram for Carrier

Table 3: Carrier Database Relationship Table

Sr. No	Table Name	Field Name	Key Type	Table Name	Field Name	Key Type	Relationship
1.	CarrierShipmentTransaction	State	Secondary	States	State	Primary	One-One
2.	CarrierShipmentTransaction	UnitWeightMeasure	Secondary	CarrierPrice	Weight	Secondary	One-One
3.	CarrierShipmentTransaction	ShipFromPartyId	Secondary	CarrierSupplier	SupplierId	Primary	One-One
4.	CarrierShipmentTransaction	ShipFromPartyId	Secondary	CarrierManufacturer	MfrId	Primary	One-One
5.	CarrierShipmentTransaction	ShipToPartyId	Secondary	CarrierRetailerTable	RetailerId	Primary	One-One
6.	CarrierShipmentTransaction	ShipToPartyId	Secondary	CarrierManufacturer	MfrId	Primary	One-One

Sr. No.	TableName	FieldName	Allow Nulls?	DataType
1	CarrierShipmentTransaction			
		DocumentID	NO	nchar(25)
		DocumentDateTime	NO	datetime
		ShipUnitQuantity	NO	decimal(18, 2)
		Address1	NO	char(100)
		City	YES	char(25)
		State	YES	nchar(20)
		[Postal Code]	YES	nchar(25)
		UnitWeightMeasure	NO	decimal(18, 0)
		UnitVolumeMeasure	NO	decimal(18, 0)
		ShipFromPartyID	NO	char(50)
		ShipToPartyID	NO	char(50)
		ShipmentTotalAmountRoad	NO	money
		ShipmentTotalAmountAir	NO	money
		PromisedDeliveryDateTimeRoad	YES	datetime
		PromisedDeliveryDateTimeAir	YES	datetime
		ShippedQuantity	YES	int
		PurchaseOrderReference	NO	nchar(25)
		Status	NO	char(25)
		FinalModeOfTransport	YES	nchar(10)
		ProductId	NO	nchar(10)
2	**CarrierSupplier**			
		SupplierId	NO	char(50)
		SupplierName	NO	Char(50)
		Address1	NO	Char(100)
		City	NO	char(50)
		State	NO	Char(50)
		[Postal Code]	NO	Int
		ContactName	NO	Char(50)
		ContactTel	NO	Char(50)

		WebAddress	YES	char(50)
		Email	YES	char(50)
		LastTransactionDate	YES	datetime
		FirstTransactionDate	YES	datetime
		Rating	NO	nchar(25)
3	**CarrierManufacturer**			
		MfrId	NO	char(50)
		MfrName	NO	char(30)
		Address1	NO	Char(100)
		City	NO	char(30)
		State	NO	char(30)
		PostalCode	NO	nchar(20)
		ContactName	NO	char(30)
		ContactTel	NO	char(30)
		WebAddress	YES	char(50)
		Email	NO	char(50)
		LastTransactionDate	YES	datetime
		FirstTransactionDate	YES	datetime
		Rating	NO	char(30)
4	**CarrierPrice**			
		SerialNumber	NO	Int
		Weight	NO	decimal(18, 2)
		Volume	NO	decimal(18, 2)
		[Road Price]	NO	decimal(18, 2)
		[Air Price]	NO	decimal(18, 2)
5	**CarrierTransportationEquipment**			
		TransportVehicleId	NO	nchar(10)
		TypeOfVehicle	NO	char(10)
		Capacity	NO	int
		NextAvailableDate	YES	datetime
		Available	NO	bit

APPENDIX 5

CODE EXECUTION

This section provides a step by step instruction to the user on setting up and running the different programs in the thesis. It explains which programs to start, in what sequence and how to carry out different operations which the programs support. Every program requires the SQL Server 2005 Express Management studio, which is used to develop the databases, to be running. For this thesis, the management studio is on "Machine 9P3MCB1" in the ASI lab. The computers used to run the programs and the SQL server should be in the same network.

Setting up the Supplier agent:

1) Run the "Supplier_Database_2" web service in the "Run without debugging mode".
2) Run the "Supplier_KnowledgeSource_New" web service in the "Run without debugging mode".
3) Run the "MAC_Supplier_New" web service in the "Run without debugging mode".
4) From the "MAC_Supplier_New" web service add web reference to the database web service by right clicking on "App_WebReferences" in the explorer window and then select "Add Web Reference...". Select the URL of the database web service and copy it into the URL field of "Add Web Reference" screen and click "Go". Type "Supplier_Database" in the "Web reference name" field on the screen and click the "Add Reference" button.
5) Similarly, add the knowledge source web service reference and type "Supplier_KnowledgeSource" in the "Web reference name" field and click the "Add Reference" button.
6) Now run the "MAC_Supplier_New" web service.
7) Now run the "Thesis_New" project and add the web reference to the MAC web service which was setup in step 6. Type "Supplier_MAC_New" in the "Web reference name" field on the "Add Web Reference" screen.
8) Now the user can perform different operation for the supplier.

If the web references are already setup, then the user should run the web services in the above sequence and instead of adding the reference the user should only do update references by right clicking on the added web references and then selecting "Update Web Reference".

If the user wants to run the Thesis_New project on a machine different from the one running the web services then the "MAC_Supplier_New" web service should be setup accordingly. Internet Information Service needs to be setup for the same.

Steps to setup the Internet Information Service to run the Thesis_New project on a different file:

You need to have administrative previledges to perform this operations.

1) Open Control Panel and click on Administrative Tools
2) In Administrative Tools open Internet Information Services (IIS)
3) In the Internet Information Service window expand the "Computer Name (local computer)" section. Then open the Web Sites section.
4) Right click on Default Web Site section and select "Virtual Directory" under the "New" tab.
5) Provide an Alias name of your choice.
6) In the next step browse to the directory where your have stored the "service.asmx" file of the web service that you want to access remotely. For example the address to the MAC_Supplier_New web service is "C:\Documents and Settings\Vilesh\My Documents\Visual Studio 2005\WebSites\MAC_Supplier_New".
7) In the next screen select the Read, Run Scripts (such as ASP), Execute (such as IASPI applications or CGI) and Browse.
8) Click Next and then Finish. This creates a virtual directory for the web services.
9) Right click on the alias which you just created and select properties.
10) Under the virtual directory tab select "Script source access". Under the ASP.Net tab, select ASP.Net version as "2.0.50727" and click Apply and then OK.
11) Browse to the folder which was selected above. Right click and select properties. Under the security tab, add "NETWORK SERVICE" and "EVERYONE" as new group users.

Now you are ready to access the web service remotely. On the remote computer open a new internet explorer window. In the url field type http://(IP address of the computer running the

web service)/alias/Service.asmx. This should show you the web methods of the web service which you are accessing remotely. This url should be provided as the web reference.

Note: The steps mentioned above can change depending on your network settings and version of IIS you are using.

Setting up the Manufacturer agent:

1) Run the "Database_MAC" web service in the "Run without debugging mode".
2) Run the "KnowledgeSource_Manufacturer" web service in the "Run without debugging mode".
3) Run the "MAC_Manufacturer" web service in the "Run without debugging mode".
4) From the "MAC_Manufacturer" web service add web reference to the database web service by right clicking on "App_WebReferences" in the explorer window and then select "Add Web Reference...". Select the URL of the database web service and copy it into the URL field of "Add Web Reference" screen and click "Go". Type "Database_Manufacturer" in the "Web reference name" field on the screen and click the "Add Reference" button.
5) Similarly, add the knowledge source web service reference and type "KnowledgeSource_Manufacturer" in the "Web reference name" field and click the "Add Reference" button.
6) Now run the "MAC_Manufacturer" web service.
7) Now run the "Thesis_New_Manufacturer project and add the web reference to the MAC web service which was setup in step 6. Type "MAC_Manufacturer" in the "Web reference name" field on the "Add Web Reference" screen.
8) Now the user can perform different operation for the manufacturer.

Similarly, if the references are already setup, then the user should run the web services in the above sequence and instead of adding the references the user should only do update references by right clicking on the added web reference and then selecting "Update Web Reference". If the user wants to run the web services and the main project on different computers then follow procedure similar to one explained for the supplier.

Setting up the Carrier Agent:

1) Run the "Database_Carrier" web service in the "Run without debugging mode".
2) Run the "KnowledgeSource_Carrier" web service in the "Run without debugging mode".
3) Run the "MAC_Carrier_NEW" web service in the "Run without debugging mode".
4) From the "MAC_Carrier_NEW" web service add web reference to the database web service by right clicking on "App_WebReferences" in the explorer window and then select "Add Web Reference...". Select the URL of the database web service and copy it into the URL field of "Add Web Reference" screen and click "Go". Type "Database_Carrier" in the "Web reference name" field on the screen and click the "Add Reference" button.
5) Similarly, add the knowledge source web service reference and type "KnowledgeSource_Carrier" in the "Web reference name" field and click the "Add Reference" button.
6) Now run the "MAC_Carrier_NEW" web service.
7) Now run the "Thesis_New_Carrier" project and add the web reference to the MAC web service which was setup in step 6. Type "MAC_Carrier" in the "Web reference name" field on the "Add Web Reference" screen.
8) Now the user can perform different operation for the Carrier.

If the web references are already added, then the user should only update the web references. Similarly, to run the carrier programs on different machines, follow procedure similar to the one described for the supplier.

APPENDIX 6

PROGRAM CODE

This section provides the C# code which was written for this thesis. It provides the code for the supplier Agent Module. Code for manufacturer and carrier are not presented here due to their similarity with the supplier code.

Supplier Code

Supplier code is divided into four different code modules. Each individual code module refers to the code for the Supplier GUI, Supplier Module Activation Controller web service, Supplier Database Controller web service and Supplier Knowledge Source web service.

Supplier GUI code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Thesis_New.Supplier_MAC_New;

namespace Thesis_New
{
    public partial class Form1 : Form
    {
       Supplier_MAC_New.Service service = new Service();
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

        }
        private void btn_GetRFQ_Click(object sender, EventArgs e)
        {
            //generate the GetRFQ object
            GetRFQ grfq = new GetRFQ();

            //populate the quantity
            grfq.DataArea = new GetRFQDataAreaType();
            grfq.DataArea.RFQ = new RFQ[1];
            grfq.DataArea.RFQ[0] = new RFQ();
            grfq.DataArea.RFQ[0].RFQLine = new RFQLine[1];
            grfq.DataArea.RFQ[0].RFQLine[0] = new RFQLine();
            grfq.DataArea.RFQ[0].RFQLine[0].Quantity = new
Quantity();
            grfq.DataArea.RFQ[0].RFQLine[0].Quantity.Value =
Decimal.Parse(txt_Quantity.Text);

            //add the requested delivery date
            grfq.DataArea.RFQ[0].RFQLine[0].RequiredDeliveryDateTime
= monthCalendar1.SelectionStart.ToShortDateString();

            //add the retailer id
            grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty = new
RequesterParty();
            grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty.PartyIDs =
new PartyIDs();

grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty.PartyIDs.ID = new
ID[1];

grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty.PartyIDs.ID[0] = new
ID();

grfq.DataArea.RFQ[0].RFQLine[0].RequesterParty.PartyIDs.ID[0].Value =
txt_MfrId.Text;
```

```
            //add the product id
            grfq.DataArea.RFQ[0].RFQLine[0].Item = new Item();
            grfq.DataArea.RFQ[0].RFQLine[0].Item.ItemID = new
ItemID[1];
            grfq.DataArea.RFQ[0].RFQLine[0].Item.ItemID[0] = new
ItemID();
            grfq.DataArea.RFQ[0].RFQLine[0].Item.ItemID[0].ID = new
ID();
            grfq.DataArea.RFQ[0].RFQLine[0].Item.ItemID[0].ID.Value =
txt_ProductId.Text;

            ShowRFQ srfq = new ShowRFQ();
            srfq = service.ReceiveGetRFQ(grfq);

            txt_UnitPrice.Text =
srfq.DataArea.RFQ[0].RFQLine[0].UnitPrice.PerQuantity.Value.ToString(
);
            txt_RFQId.Text = srfq.ApplicationArea.BODID.Value;
            dateTimePicker1.Text =
srfq.DataArea.RFQ[0].RFQLine[0].RequiredDeliveryDateTime;
            txt_ReturnMsg.Text =
srfq.DataArea.RFQ[0].RFQLine[0].Description[0].Value;

        }


        private void btn_GetPO_Click_1(object sender, EventArgs e)
        {
            GetPurchaseOrder gpo = new GetPurchaseOrder();
            gpo.DataArea = new GetPurchaseOrderDataAreaType();
            gpo.DataArea.PurchaseOrder = new PurchaseOrder[1];
            gpo.DataArea.PurchaseOrder[0] = new PurchaseOrder();
            gpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader = new
PurchaseOrderHeader();

gpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID = new
DocumentID();

gpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID = new
ID();

gpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Value
= txt_RfqIdPO.Text;

            ShowPurchaseOrder showpo = new ShowPurchaseOrder();

            showpo = service.ReceiveGetPurchaseOrder(gpo);

            txt_POID.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Va
lue;
            txt_UnitPricePOO.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].UnitPrice.PerQu
antity.Value.ToString();
```

```
txt_QuantOffered.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Quantity.Value.
ToString();
            dateTimePicker2POO.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].RequiredDeliver
yDateTime;
            txt_ReturnMessagePO.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Description[0].
Value;
        }

        private void btn_ProcessPO_Click(object sender, EventArgs e)
        {
            ProcessPurchaseOrder procpo = new ProcessPurchaseOrder();
            procpo.DataArea = new ProcessPurchaseOrderDataAreaType();
            procpo.DataArea.PurchaseOrder = new PurchaseOrder[1];
            procpo.DataArea.PurchaseOrder[0] = new PurchaseOrder();
            procpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader =
new PurchaseOrderHeader();

procpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID = new
DocumentID();

procpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID =
new ID();

procpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Va
lue = txt_POId2.Text;
            AcknowledgePurchaseOrder ackpo = new
AcknowledgePurchaseOrder();
            ackpo = service.ReceiveProcessPO(procpo);
            txt_POID.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Val
ue;
            txt_UnitPricePOO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].UnitPrice.PerQua
ntity.Value.ToString();
            txt_QuantOffered.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Quantity.Value.T
oString();
            dateTimePicker2POO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].RequiredDelivery
DateTime;
            txt_ReturnMessagePO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Description[0].V
alue;
        }

        private void btn_CancelPO_Click(object sender, EventArgs e)
        {
            CancelPurchaseOrder cancelpo = new CancelPurchaseOrder();
            cancelpo.DataArea = new
CancelPurchaseOrderDataAreaType();
            cancelpo.DataArea.PurchaseOrder = new PurchaseOrder[1];
            cancelpo.DataArea.PurchaseOrder[0] = new PurchaseOrder();
```

```
cancelpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader = new
PurchaseOrderHeader();

cancelpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID =
new DocumentID();

cancelpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID
= new ID();

cancelpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.
Value = txt_POId2.Text;

            AcknowledgePurchaseOrder ackpo = new
AcknowledgePurchaseOrder();
            ackpo = service.ReceiveCancelPO(cancelpo);
            txt_POID.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Val
ue;
            txt_UnitPricePOO.Clear();
            //txt_UnitPriceOfferedPOOT.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].UnitPrice.PerQua
ntity.Value.ToString();
            txt_QuantOffered.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Quantity.Value.T
oString();
            dateTimePicker2POO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].RequiredDelivery
DateTime;
            txt_ReturnMessagePO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Description[0].V
alue;
        }

        private void btn_ChangePO_Click(object sender, EventArgs e)
        {
            ChangePurchaseOrder changepo = new ChangePurchaseOrder();
            changepo.DataArea = new
ChangePurchaseOrderDataAreaType();
            changepo.DataArea.PurchaseOrder = new PurchaseOrder[1];
            changepo.DataArea.PurchaseOrder[0] = new PurchaseOrder();
            changepo.DataArea.PurchaseOrder[0].PurchaseOrderHeader =
new PurchaseOrderHeader();

changepo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID =
new DocumentID();

changepo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID
= new ID();

changepo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.
Value = txt_POId2.Text;
            changepo.DataArea.PurchaseOrder[0].PurchaseOrderLine =
new PurchaseOrderLine[1];
            changepo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0] =
new PurchaseOrderLine(); ;
```

```
changepo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].RequiredDeliv
eryDateTime = monthCalendar2PO.SelectionStart.ToShortDateString();

changepo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Quantity =
new Quantity();

changepo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Quantity.Valu
e = decimal.Parse(txt_QuantityPO.Text);

            AcknowledgePurchaseOrder ackpo = new
AcknowledgePurchaseOrder();
            ackpo = service.ReceiveChangePO(changepo);

            txt_POID.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Val
ue;
            txt_UnitPricePOO.Clear();
            //txt_UnitPriceOfferedPOOT.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].UnitPrice.PerQua
ntity.Value.ToString();
            txt_QuantOffered.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Quantity.Value.T
oString();
            dateTimePicker2POO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].RequiredDelivery
DateTime;
            txt_ReturnMessagePO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Description[0].V
alue;
        }

        private void splitContainer1_Panel1_Paint(object sender,
PaintEventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void txt_ProductId_TextChanged(object sender,
EventArgs e)
        {

        }

        private void txt_Quantity_TextChanged(object sender,
EventArgs e)
        {

        }
```

```
private void monthCalendar1_DateChanged(object sender,
DateRangeEventArgs e)
        {

        }

        private void label6_Click(object sender, EventArgs e)
        {

        }

        private void txt_MfrId_TextChanged(object sender, EventArgs
e)
        {

        }

        private void label5_Click(object sender, EventArgs e)
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void btn_ChangeRFQ_Click(object sender, EventArgs e)
        {
            ChangeRFQ chrfq = new ChangeRFQ();
            chrfq.ApplicationArea = new ApplicationArea();
            chrfq.ApplicationArea.BODID = new BODID();
            chrfq.ApplicationArea.BODID.Value = txt_RFQIDinput.Text;

            chrfq.DataArea = new ChangeRFQDataAreaType();
            chrfq.DataArea.RFQ = new RFQ[1];
            chrfq.DataArea.RFQ[0] = new RFQ();
            chrfq.DataArea.RFQ[0].RFQLine = new RFQLine[1];
            chrfq.DataArea.RFQ[0].RFQLine[0] = new RFQLine();
            chrfq.DataArea.RFQ[0].RFQLine[0].UnitPrice = new
UnitPrice();
            chrfq.DataArea.RFQ[0].RFQLine[0].UnitPrice.PerQuantity =
new PerQuantity();

chrfq.DataArea.RFQ[0].RFQLine[0].UnitPrice.PerQuantity.Value =
decimal.Parse(txt_ReqUnitPrice.Text);

            AcknowledgeRFQ ackrfq = new AcknowledgeRFQ();

            ackrfq = service.ReceiveChangeRFQ(chrfq);
            txt_UnitPrice.Text =
ackrfq.DataArea.RFQ[0].RFQLine[0].UnitPrice.PerQuantity.Value.ToStrin
g();
```

```
 txt_ReturnMsg.Text =
ackrfq.DataArea.RFQ[0].RFQLine[0].Description[0].Value;
            //txt_RFQId.Text = txt_RFQIDinput.Text;
         }

        private void btn_GetPurchaseOrder_Click(object sender,
EventArgs e)
        {
            GetPurchaseOrder gpo = new GetPurchaseOrder();
            gpo.DataArea = new GetPurchaseOrderDataAreaType();
            gpo.DataArea.PurchaseOrder = new PurchaseOrder[1];
            gpo.DataArea.PurchaseOrder[0] = new PurchaseOrder();
            gpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader = new
PurchaseOrderHeader();

gpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID = new
DocumentID();

gpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID = new
ID();

gpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Value
= txt_RfqIdPO.Text;

            ShowPurchaseOrder showpo = new ShowPurchaseOrder();
            showpo = service.ReceiveGetPurchaseOrder(gpo);

            txt_POID.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Va
lue;
            txt_UnitPricePOO.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].UnitPrice.PerQu
antity.Value.ToString();
            txt_QuantOffered.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Quantity.Value.
ToString();
            dateTimePicker2POO.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].RequiredDeliver
yDateTime;
            txt_ReturnMessagePO.Text =
showpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Description[0].
Value;
        }

        private void btn_AcceptPurchaseOrder_Click(object sender,
EventArgs e)
        {
            ProcessPurchaseOrder procpo = new ProcessPurchaseOrder();
            procpo.DataArea = new ProcessPurchaseOrderDataAreaType();
            procpo.DataArea.PurchaseOrder = new PurchaseOrder[1];
            procpo.DataArea.PurchaseOrder[0] = new PurchaseOrder();
            procpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader =
new PurchaseOrderHeader();
```

```
procpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID = new
DocumentID();

procpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID =
new ID();

procpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Va
lue = txt_POId2.Text;

            AcknowledgePurchaseOrder ackpo = new
AcknowledgePurchaseOrder();
            ackpo = service.ReceiveProcessPO(procpo);
            txt_POID.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Val
ue;
            txt_UnitPricePOO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].UnitPrice.PerQua
ntity.Value.ToString();
            txt_QuantOffered.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Quantity.Value.T
oString();
            dateTimePicker2POO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].RequiredDelivery
DateTime;
            txt_ReturnMessagePO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Description[0].V
alue;

        }

        private void btn_CancelPurchaseOrder_Click(object sender,
EventArgs e)
        {
            CancelPurchaseOrder cancelpo = new CancelPurchaseOrder();
            cancelpo.DataArea = new
CancelPurchaseOrderDataAreaType();
            cancelpo.DataArea.PurchaseOrder = new PurchaseOrder[1];
            cancelpo.DataArea.PurchaseOrder[0] = new PurchaseOrder();
            cancelpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader =
new PurchaseOrderHeader();

cancelpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID =
new DocumentID();

cancelpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID
= new ID();

cancelpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.
Value = txt_POId2.Text;

            AcknowledgePurchaseOrder ackpo = new
AcknowledgePurchaseOrder();
            ackpo = service.ReceiveCancelPO(cancelpo);
```

```
txt_POID.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderHeader.DocumentID.ID.Val
ue;
            txt_UnitPricePOO.Clear();
            //txt_UnitPriceOfferedPOOT.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].UnitPrice.PerQua
ntity.Value.ToString();
            txt_QuantOffered.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Quantity.Value.T
oString();
            dateTimePicker2POO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].RequiredDelivery
DateTime;
            txt_ReturnMessagePO.Text =
ackpo.DataArea.PurchaseOrder[0].PurchaseOrderLine[0].Description[0].V
alue;
        }

        private void purchase_OrderToolStripButton_Click(object
sender, EventArgs e)
        {
            try
            {

this.supplierPurchaseOrderDataTableAdapter.Purchase_Order(this.suppli
er_VileshDataSet.SupplierPurchaseOrderData);
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }
        }

        private void rFQ_DataToolStripButton_Click(object sender,
EventArgs e)
        {
            try
            {

this.supplierRFQDataTableAdapter.RFQ_Data(this.supplier_VileshDataSet
1.SupplierRFQData);
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }
        }

        private void shop_Floor_DataToolStripButton_Click(object
sender, EventArgs e)
        {
            try
            {
```

```
this.supplierShopFloorDataTableAdapter.Shop_Floor_Data(this.supplier_
VileshDataSet2.SupplierShopFloorData);
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }
        }
    }
}
```

Supplier Module Activation Controller Code:

```
//Module Activation Controller for the Manufacturer
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using Vilesh_Thesis;

[WebService(Namespace = "http://www.rit.edu/~asi")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    Supplier_Database.Service servicedatabase = new
Supplier_Database.Service();
    Supplier_KnowledgeSource.Service serviceks = new
Supplier_KnowledgeSource.Service();

    public Service()
    {
        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    //method to receive GetRFQ
    [WebMethod]
    public ShowRFQ ReceiveGetRFQ(GetRFQ getrfq)
    {
        string mfrid, prodid, rfqid, messagerfq, priceupdate;
        DateTime reqdd, EarliestShipDate;
        decimal reqquant, AvailableQuant, sum = 0, linedupprod = 0,
ModifiedunitPrice;
        bool rfq;
        string[] ProdCap, ProdCur, Data;
        int availabledays, actualavailabledays, carrierdays = 3;

        RFQ MyRFQ = new RFQ();
        RFQLine myRFQLine = new RFQLine();
        ItemID myItemId = new ItemID();
        ID myId = new ID();
```

```
        System.Collections.ArrayList myArraylist =
getrfq.DataArea.RFQCollection;
        System.Collections.IEnumerator myEnum =
myArraylist.GetEnumerator();

        while (myEnum.MoveNext())
        {
            MyRFQ = (RFQ)myEnum.Current;
        }

        //myEnum = myRFQLine.Item.ItemIDCollection.GetEnumerator();
        myEnum = MyRFQ.RFQLineCollection.GetEnumerator();

        while (myEnum.MoveNext())
        {
            myRFQLine = (RFQLine)myEnum.Current;
        }

        //get the required quantity data
        reqquant = myRFQLine.Quantity.MixedValue;

        //calculate the required Delivery date
        reqdd = DateTime.Parse(myRFQLine.RequiredDeliveryDateTime);

        myEnum = myRFQLine.Item.ItemIDCollection.GetEnumerator();

        while (myEnum.MoveNext())
        {
            myItemId = (ItemID)myEnum.Current;
        }

        //retrieve the Product Id
        prodid = myItemId.ID.MixedValue;

        //retreive the RFQId.
        rfqid = "RFQ" + (new Random()).Next(100000).ToString();

        //retreive retailer ID.
        myEnum =
myRFQLine.RequesterParty.PartyIDs.IDCollection.GetEnumerator();

        while (myEnum.MoveNext())
        {
            myId = (ID)myEnum.Current;
        }

        mfrid = myId.MixedValue;

        //place a call to database ws to add data to
ManufacturerRFQData table.
        servicedatabase.UpdateRFQData(mfrid, prodid, reqdd, reqquant,
rfqid, "false");

        //check if the required quantity is currently available in
stock
```

```
        AvailableQuant =
servicedatabase.CheckProductAvailability(prodid);

        //retrieve unit Price information
        Data = servicedatabase.GetUnitPrice(prodid);

        if (AvailableQuant >= reqquant)
        {
            //decide unit price
            //make a call to KS WS
            rfq = true;
            ModifiedunitPrice =
serviceks.DecideUnitPrice(AvailableQuant, reqquant, Data, rfq);
            messagerfq = "Required Quantity is available";
            EarliestShipDate = reqdd;
        }
        else
        {
            //Retrive Available Current Capacity
            ProdCap =
servicedatabase.currentProductionCapacity(prodid);

            //this gives the production capacity for a product per
day
            for (int i = 0; i < ProdCap.Length; i++)
            {
                sum = sum + Decimal.Parse(ProdCap[i]);
            }

            //this gives the current Production linedup
            ProdCur = servicedatabase.currentProduction(prodid);
            for (int a = 0; a < ProdCur.Length; a++)
            {
                linedupprod = linedupprod +
Decimal.Parse(ProdCur[a]);
            }

            //number of days between required delivery date and
current date
            DateTime availabledaysinter = System.DateTime.Now;
            availabledays = reqdd.Day - availabledaysinter.Day;

            // assuming that it will take 3 days for carrier to
transport goods . This is assumption
            //only for RFQ purpose

            actualavailabledays = availabledays - carrierdays;

            if (((actualavailabledays * sum) - reqquant) >
linedupprod)
            {
                rfq = true;

                //decide the unitprice.
                //make a method call to KS WS
```

```
 ModifiedunitPrice = serviceks.DecideUnitPrice(AvailableQuant,
reqquant, Data, rfq);



 messagerfq = "Required quantity can be manufactured by requested
Delivery Date";
                EarliestShipDate = reqdd;
            }
            else
            {
                rfq = false;
                //calculate the reduced price we are willing to offer
                ModifiedunitPrice =
serviceks.DecideUnitPrice(AvailableQuant, reqquant, Data, rfq);

                ////calculate the earliest possible delivery date
                EarliestShipDate =
serviceks.EarliestPossibleDeliveryDate(linedupprod, sum, reqquant,
carrierdays);
                messagerfq = "Required quantity cannot be
manufactured by requested Delivery Date, possible delivery date is";
            }
        }

        priceupdate = "false";

        //update the offered price and Promised Shipment Date data in
ManufacturerRFQData Table
        servicedatabase.UpdateOfferedPriceData(rfqid,
ModifiedunitPrice, EarliestShipDate, priceupdate);

        //preparing the ShowRFQ
        ShowRFQ srfq = new ShowRFQ();
        RFQ srfqmyRFQ = new RFQ();
        RFQLine srfqmyRFQLine = new RFQLine();
        srfqmyRFQLine.UnitPrice.PerQuantity.MixedValue =
ModifiedunitPrice;
        srfqmyRFQLine.RequiredDeliveryDateTime =
EarliestShipDate.ToShortDateString();
        Description srfqmyDescription = new Description();
        srfqmyDescription.MixedValue = messagerfq;
        srfqmyRFQLine.DescriptionCollection.Add(srfqmyDescription);
        srfqmyRFQ.RFQLineCollection.Add(srfqmyRFQLine);
        srfq.DataArea.RFQCollection.Add(srfqmyRFQ);
        //srfqmyRFQHeader.DescriptionCollection.
        srfq.ApplicationArea.BODID.MixedValue = rfqid;

        return (srfq);
    }


    //method to retreive ChangeRFQ
    [WebMethod]
    public AcknowledgeRFQ ReceiveChangeRFQ(ChangeRFQ chrfq)
    {
```

```
        string rfqid, mfrid, prodid, statusupdate, mfrstatus,
return_message;
        string[] data;

       decimal unitprice, reqquant, unitprice_old, new_price;
        DateTime promisedshipdate;

        System.Collections.ArrayList myArraylist =
chrfq.DataArea.RFQCollection;
        System.Collections.IEnumerator myEnum =
myArraylist.GetEnumerator();

        RFQ myRFQ = new RFQ();
        while (myEnum.MoveNext())
        {
            myRFQ = (RFQ)myEnum.Current;
        }

        myEnum = myRFQ.RFQLineCollection.GetEnumerator();

        RFQLine myRFQLine = new RFQLine();

        while (myEnum.MoveNext())
        {
            myRFQLine = (RFQLine)myEnum.Current;
        }

        rfqid = chrfq.ApplicationArea.BODID.MixedValue;
        unitprice = myRFQLine.UnitPrice.PerQuantity.MixedValue;

        //retreive data from the RFQ table for the price offered.
        data = servicedatabase.RetreivePriceUpdateRFQ(rfqid);

        mfrid = data[0];
        unitprice_old = decimal.Parse(data[1]);
        statusupdate = data[2].Trim();
        reqquant = decimal.Parse(data[3]);
        promisedshipdate = DateTime.Parse(data[4]);

        //retreive the manufacturer status from ManufacturerRetailer
table
        mfrstatus = servicedatabase.RetreiveStatusInfo(mfrid);

        if (statusupdate == "false")
        {
            string priceupdate = "true";

            //calculate the new price to be offered and update the
rfq table
            new_price = serviceks.DecideUpForChangeRequest(mfrstatus,
unitprice_old);
            return_message = "Congratulations, Your Updated Price is
as displayed";
            //servicedatabase.UpdateRFQPriceUpdateStatus(rfqid,
new_price);
```

```
//update the offered price and Promised Shipment Date data in
SupplierRFQData Table
            servicedatabase.UpdateOfferedPriceData(rfqid, new_price,
promisedshipdate, priceupdate);
        }
        else
        {
            new_price = unitprice_old;
            return_message = "Sorry, your price cannot be changed
further";
        }

        AcknowledgeRFQ ackrfq = new AcknowledgeRFQ();
        RFQ ackrfqmyRFQ = new RFQ();
        RFQLine ackrfqmyRFQLine = new RFQLine();
        ackrfqmyRFQLine.UnitPrice.PerQuantity.MixedValue = new_price;
        //ackrfqmyRFQLine.RequiredDeliveryDateTime =
EarliestShipDate.ToShortDateString();
        Description ackrfqmyDescription = new Description();
        ackrfqmyDescription.MixedValue = return_message;

ackrfqmyRFQLine.DescriptionCollection.Add(ackrfqmyDescription);

        ackrfqmyRFQ.RFQLineCollection.Add(ackrfqmyRFQLine);

        ackrfq.DataArea.RFQCollection.Add(ackrfqmyRFQ);
        //srfqmyRFQHeader.DescriptionCollection.

        return (ackrfq);
    }

    //method to receive getPurchaseOrder
    [WebMethod]
    public ShowPurchaseOrder ReceiveGetPurchaseOrder(GetPurchaseOrder
getpo)
    {
        string rfqid, poid, returnmessage;
        string[] Data;

        System.Collections.ArrayList myArraylist =
getpo.DataArea.PurchaseOrderCollection;
        System.Collections.IEnumerator myEnum =
myArraylist.GetEnumerator();

        PurchaseOrder myPurchaseOrder = new PurchaseOrder();
        while (myEnum.MoveNext())
        {
            myPurchaseOrder = (PurchaseOrder)myEnum.Current;
        }

        rfqid =
myPurchaseOrder.PurchaseOrderHeader.DocumentID.ID.MixedValue;

        poid = "PO" + (new Random()).Next(100000).ToString();
```

```
//retreive the data related to the rfq
        Data = servicedatabase.RetreivePriceUpdateRFQ(rfqid);

        //update the purchase order
        servicedatabase.InsertPOData(poid, Data[0],
decimal.Parse(Data[1]) * decimal.Parse(Data[3]),
DateTime.Parse(Data[5]), DateTime.Parse(Data[4]),
decimal.Parse(Data[3]), "Negotiation", Data[6], Data[1], rfqid);

        returnmessage = "Your Requested Purchase Order is as shown";
        ShowPurchaseOrder showpo = new ShowPurchaseOrder();

        PurchaseOrder myPurchaseOrder2 = new PurchaseOrder();
        myPurchaseOrder2.PurchaseOrderHeader.DocumentID.ID.MixedValue
= poid;

        Description myDescription = new Description();
        myDescription.MixedValue = returnmessage;

        PurchaseOrderLine myPurchaseOrderLine = new
PurchaseOrderLine();
        myPurchaseOrderLine.UnitPrice.PerQuantity.MixedValue =
decimal.Parse(Data[1]);
        myPurchaseOrderLine.Quantity.MixedValue =
decimal.Parse(Data[3]);
        myPurchaseOrderLine.RequiredDeliveryDateTime = Data[4];

        myPurchaseOrderLine.DescriptionCollection.Add(myDescription);


myPurchaseOrder2.PurchaseOrderLineCollection.Add(myPurchaseOrderLine)
;


showpo.DataArea.PurchaseOrderCollection.Add(myPurchaseOrder2);

        return showpo;
    }

    //method to receive ProcessPO
    [WebMethod]
    public AcknowledgePurchaseOrder
ReceiveProcessPO(ProcessPurchaseOrder processpo)
    {
        string poid, returnmessage;
        string[] Data, Data_CP;
        decimal newprod;

        System.Collections.ArrayList myArraylist =
processpo.DataArea.PurchaseOrderCollection;
        System.Collections.IEnumerator myEnum =
myArraylist.GetEnumerator();

        PurchaseOrder myPurchaseOrder = new PurchaseOrder();
        while (myEnum.MoveNext())
        {
```

```
    myPurchaseOrder = (PurchaseOrder)myEnum.Current;
        }

        poid =
myPurchaseOrder.PurchaseOrderHeader.DocumentID.ID.MixedValue;

        //retreive the data related to purchase order
        Data = servicedatabase.RetreivePurchaseOrderData(poid);

        //change the status from Negotiation to Complete
        servicedatabase.UpdateStatus(poid);

        returnmessage = "Congratulations, Your Purchase Order has
been successfully completed";

        //update the shop floor data
        Data_CP = servicedatabase.currentProduction(Data[3]);
        newprod = decimal.Parse(Data_CP[0]) + decimal.Parse(Data[6]);
        servicedatabase.UpdateShopFloorData(Data[3], newprod);


        //prepare the object to be sent back
        AcknowledgePurchaseOrder acknpo = new
AcknowledgePurchaseOrder();

        PurchaseOrder myPurchaseOrder2 = new PurchaseOrder();
        myPurchaseOrder2.PurchaseOrderHeader.DocumentID.ID.MixedValue
= poid;

        Description myDescription = new Description();
        myDescription.MixedValue = returnmessage;

        PurchaseOrderLine myPurchaseOrderLine = new
PurchaseOrderLine();
        myPurchaseOrderLine.UnitPrice.PerQuantity.MixedValue =
decimal.Parse(Data[2]);
        myPurchaseOrderLine.Quantity.MixedValue =
decimal.Parse(Data[1]);
        myPurchaseOrderLine.RequiredDeliveryDateTime = Data[0];

        myPurchaseOrderLine.DescriptionCollection.Add(myDescription);


myPurchaseOrder2.PurchaseOrderLineCollection.Add(myPurchaseOrderLine)
;

acknpo.DataArea.PurchaseOrderCollection.Add(myPurchaseOrder2);

        return acknpo;
    }

    //method to receive changepo
    [WebMethod]
    public AcknowledgePurchaseOrder
```

```
ReceiveChangePO(ChangePurchaseOrder changepo)
    {


 string poid, postatus = " ", returnmessage, mfrstatus, priceupdate,
messagerfq = " ", mfrid, prodid, rfqid;
        string[] Data_PO, Data_UnitPrice, ProdCap, ProdCur;
        DateTime reqdd, EarliestShipDate = System.DateTime.Now;
        decimal newprice, reqquant, ModifiedunitPrice, finaloldprice,
AvailableQuant, sum = 0, linedupprod = 0;
        bool rfq;
        int availabledays, actualavailabledays, carrierdays = 3;

        System.Collections.ArrayList myArraylist =
changepo.DataArea.PurchaseOrderCollection;
        System.Collections.IEnumerator myEnum =
myArraylist.GetEnumerator();

        PurchaseOrder myPurchaseOrder = new PurchaseOrder();
        while (myEnum.MoveNext())
        {
            myPurchaseOrder = (PurchaseOrder)myEnum.Current;
        }

        poid =
myPurchaseOrder.PurchaseOrderHeader.DocumentID.ID.MixedValue;
        myEnum =
myPurchaseOrder.PurchaseOrderLineCollection.GetEnumerator();

        PurchaseOrderLine myPurchaseOrderLine = new
PurchaseOrderLine();
        while (myEnum.MoveNext())
        {
            myPurchaseOrderLine = (PurchaseOrderLine)myEnum.Current;
        }

        reqquant = myPurchaseOrderLine.Quantity.MixedValue;
        reqdd =
DateTime.Parse(myPurchaseOrderLine.RequiredDeliveryDateTime);

        //check if the order has been completed, if not completed
then offer the change.
        //if completed then we do not offer the change
        postatus = servicedatabase.RetreivePOStatus(poid).Trim();

        //retreive data related to puchase order
        Data_PO = servicedatabase.RetreivePurchaseOrderData(poid);

        if (postatus == "Negotiation")
        {
            //logic to calculate the response to the purchase order
            //see if the required quantity reduces the cost. If it
reduces then we do not offer the change
            //if it increases the cost then we offer the change.
            //retreive unit price
```

```
  newprice =
serviceks.DecideUnitPrice(decimal.Parse((double.Parse(Data_PO[2]) +
1).ToString()), decimal.Parse(Data_PO[2]), Data_UnitPrice, true);

            //check if the required quantity is currently available
in stock

            AvailableQuant =
servicedatabase.CheckProductAvailability(Data_PO[3]);

            if (AvailableQuant >= reqquant)
            {
                //decide unit price
                //make a call to KS WS
                rfq = true;
                ModifiedunitPrice =
serviceks.DecideUnitPrice(AvailableQuant, reqquant, Data_UnitPrice,
rfq);
                messagerfq = "Required Quantity is available";
                EarliestShipDate = reqdd;

            }
            else
            {
                //Retrive Available Current Capacity
                ProdCap =
servicedatabase.currentProductionCapacity(Data_PO[3]);

                //this gives the production capacity for a product
per day
                for (int i = 0; i < ProdCap.Length; i++)
                {
                    sum = sum + Decimal.Parse(ProdCap[i]);
                }

                //this gives the current Production linedup
                ProdCur =
servicedatabase.currentProduction(Data_PO[3]);
                for (int a = 0; a < ProdCur.Length; a++)
                {
                    linedupprod = linedupprod +
Decimal.Parse(ProdCur[a]);
                }

                //number of days between required delivery date and
current date
                DateTime availabledaysinter = System.DateTime.Now;
                availabledays = reqdd.Day - availabledaysinter.Day;

                // assuming that it will take 3 days for carrier to
transport goods . This is assumption
                //only for RFQ purpose

                actualavailabledays = availabledays - carrierdays;
```

```
  if (((actualavailabledays * sum) - reqquant) > linedupprod)
                {
                    rfq = true;

                    //decide the unitprice.
                    //make a method call to KS WS
                    ModifiedunitPrice =
serviceks.DecideUnitPrice(AvailableQuant, reqquant, Data_UnitPrice,
rfq);
                    messagerfq = "Required quantity can be
manufactured by requested Delivery Date";
                    EarliestShipDate = reqdd;
                }
                else
                {
                    rfq = false;
                    //calculate the reduced price we are willing to
offer
                    ModifiedunitPrice =
serviceks.DecideUnitPrice(AvailableQuant, reqquant, Data_UnitPrice,
rfq);

                    ////calculate the earliest possible delivery date
                    EarliestShipDate =
serviceks.EarliestPossibleDeliveryDate(linedupprod, sum, reqquant,
carrierdays);
                    messagerfq = "Required quantity cannot be
manufactured by requested Delivery Date, possible delivery date is";
                }
            }


            //now we know the unit price
            //now check if the price was updated


            //retreive status message for price negotiation from RFQ
table

            // mfrstatus =
servicedatabase.RetreiveStatusInfo(Data_PO[4]);
            //priceupdate =
servicedatabase.RetreivePriceUpdateRFQ(Data_PO[5]);

            if (ModifiedunitPrice >= decimal.Parse(Data_PO[2]))
            {
                //we can change the purchase order

                messagerfq = "Your PurchaseOrder can be changed and "
+ messagerfq;
            }
            else
            {
                //we cannot change the purchase order
```

```
   messagerfq = " Your Purchase Order cannot be changed";
            }

            //if (priceupdate.Trim() = "false")
            //{
            //    //newprice =
serviceks.DecideUnitPrice(double.Parse(Data_PO[2]) + 1,
decimal.Parse(Data_PO[2]), Data_UnitPrice, true);
            //    finaloldprice = ModifiedunitPrice;
            //}
            //else
            //    if (priceupdate.Trim() = "true")
            //    {
            //        newprice =
serviceks.DecideUnitPrice(double.Parse(Data_PO[2]) + 1,
decimal.Parse(Data_PO[2]), Data_UnitPrice, true);
            //        finaloldprice =
serviceks.DecideUpForChangeRequest(mfrstatus, ModifiedunitPrice);
            //    }


            //update the ManufacturerPurchaseOrderData table with the
new Data
            servicedatabase.UpdatePOData(poid, ModifiedunitPrice *
reqquant, reqdd, EarliestShipDate, reqquant, ModifiedunitPrice);
        }
        else
        {
            messagerfq = "You cannot change the Purchase Order " +
poid;
            reqquant = decimal.Parse(Data_PO[1]);
            EarliestShipDate = DateTime.Parse(Data_PO[0]);
        }

        AcknowledgePurchaseOrder acknpo = new
AcknowledgePurchaseOrder();

        PurchaseOrder myPurchaseOrder2 = new PurchaseOrder();
        myPurchaseOrder2.PurchaseOrderHeader.DocumentID.ID.MixedValue
= poid;

        Description myDescription = new Description();
        myDescription.MixedValue = messagerfq;


        PurchaseOrderLine myPurchaseOrderLine2 = new
PurchaseOrderLine();
        //myPurchaseOrderLine.UnitPrice.PerQuantity.MixedValue =
decimal.Parse(Data_P);
        myPurchaseOrderLine2.Quantity.MixedValue = reqquant;
        myPurchaseOrderLine2.RequiredDeliveryDateTime =
EarliestShipDate.ToShortDateString();


myPurchaseOrderLine2.DescriptionCollection.Add(myDescription);
```

```
myPurchaseOrder2.PurchaseOrderLineCollection.Add(myPurchaseOrderLine2
);

acknpo.DataArea.PurchaseOrderCollection.Add(myPurchaseOrder2);

        return acknpo;
    }

    //method to receive cancelpo
    [WebMethod]
    public AcknowledgePurchaseOrder
ReceiveCancelPO(CancelPurchaseOrder cancelpo)
    {
        string poid, Data_P;
        string[] Data_PO,Data_CP;
        decimal newprod;

        System.Collections.ArrayList myArraylist =
cancelpo.DataArea.PurchaseOrderCollection;
        System.Collections.IEnumerator myEnum =
myArraylist.GetEnumerator();

        PurchaseOrder myPurchaseOrder = new PurchaseOrder();
        while (myEnum.MoveNext())
        {
            myPurchaseOrder = (PurchaseOrder)myEnum.Current;
        }

        poid =
myPurchaseOrder.PurchaseOrderHeader.DocumentID.ID.MixedValue;

        //retreive data related to the purchase order
        Data_PO = servicedatabase.RetreivePurchaseOrderData(poid);

        //do penalty calculations
        Data_P =
serviceks.PenaltyCalculation(DateTime.Parse(Data_PO[0]),
decimal.Parse(Data_PO[2]));

        //update the shop floor data
        Data_CP = servicedatabase.currentProduction(Data_PO[3]);
        newprod = decimal.Parse(Data_CP[0]) -
decimal.Parse(Data_PO[6]);
        servicedatabase.UpdateShopFloorData(Data_PO[3], newprod);

        AcknowledgePurchaseOrder acknpo = new
AcknowledgePurchaseOrder();

        PurchaseOrder myPurchaseOrder2 = new PurchaseOrder();
        myPurchaseOrder2.PurchaseOrderHeader.DocumentID.ID.MixedValue
= poid;

        Description myDescription = new Description();
        myDescription.MixedValue = Data_P;
```

```
            //update MantufacturerPurchaseOrderData
            servicedatabase.UpdateStatusCancelled(poid);

            PurchaseOrderLine myPurchaseOrderLine = new
        PurchaseOrderLine();
            //myPurchaseOrderLine.UnitPrice.PerQuantity.MixedValue =
        decimal.Parse(Data_P);
            myPurchaseOrderLine.Quantity.MixedValue =
        decimal.Parse(Data_PO[1]);
            myPurchaseOrderLine.RequiredDeliveryDateTime = Data_PO[0];

            myPurchaseOrderLine.DescriptionCollection.Add(myDescription);


        myPurchaseOrder2.PurchaseOrderLineCollection.Add(myPurchaseOrderLine)
        ;

        acknpo.DataArea.PurchaseOrderCollection.Add(myPurchaseOrder2);

            return acknpo;
        }
    }
```

Supplier Database Controller webservice code:

```
//this is the database webservice implementing database functionality
for supplier

using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Data.SqlClient;
using System.Collections.Generic;
//using Vilesh_Thesis;


[WebService(Namespace = "http://www.rit.edu/~asi/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    //SqlConnection con = new SqlConnection("Persist Security
Info=False; database=Supplier_Vilesh;server=IMERTSQLSRV;User
ID=vilesh;Pwd=52xinfra;");
    //SqlConnection con1 = new SqlConnection("Persist Security
Info=False; database=Supplier_Vilesh;server=IMERTSQLSRV;User
ID=vilesh;Pwd=52xinfra;");

    SqlConnection con = new SqlConnection("Data
Source=IMERT215\\SQLEXPRESS;Initial
Catalog=Supplier_Vilesh;Integrated Security=True");
    SqlConnection con1 = new SqlConnection("Data
Source=IMERT215\\SQLEXPRESS;Initial
Catalog=Supplier_Vilesh;Integrated Security=True");
    public Service(){
```

```
        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    public string HelloWorld()
    {
        return "Hello World";
    }

    //method to add RFQ data to SupplierRFQData Table
    [WebMethod]
    public void UpdateRFQData(string MfrId, string ProdId, DateTime
reqDelDate, decimal Quant, string docid, string priceupdatestatus)
    {
        SqlCommand cmd = new SqlCommand("INSERT INTO SupplierRFQData
(MfrId, ProductId,RequestedDeliveryDate,Quantity, RFQDocumentID,
PriceUpdateStatus) VALUES " + "(" + "@MfrId, @ProdId, @reqDelDate,
@Quant, @docid, @priceupdatestatus" + ")", con1);
        cmd.Parameters.AddWithValue("@MfrId", MfrId);
        cmd.Parameters.AddWithValue("@ProdId", ProdId);
        cmd.Parameters.AddWithValue("@reqDelDate", reqDelDate);
        cmd.Parameters.AddWithValue("@Quant", Quant);
        cmd.Parameters.AddWithValue("docid", docid);
        cmd.Parameters.AddWithValue("@priceupdatestatus",
priceupdatestatus);

        UpdateInsertDelete(cmd);
    }




    //retreive information about priceupdate
    [WebMethod]
    public string[] RetreivePriceUpdateRFQ(string rfqid)
    {
        string[] data;
        SqlCommand cmd = new SqlCommand("SELECT
MfrId,UnitPriceOffered,PriceUpdateStatus, Quantity,
PromisedDeliveryDate" + " FROM SupplierRFQData" + " WHERE
RFQDocumentId = @rfqid", con);
        cmd.Parameters.AddWithValue("@rfqid", rfqid);

         data = SelectMethod(cmd);
         return data;
    }



    [WebMethod]
    public decimal CheckProductAvailability(string ProdId)
    {
```

```
        SqlCommand cmd = new SqlCommand("SELECT AvailableQty" + "
FROM InventoryOfSupplier" + " WHERE ProductId = @ProdId", con);
        cmd.Parameters.AddWithValue("@ProdId", ProdId);

        string[] AvailableQuantity;
        AvailableQuantity = SelectMethod(cmd);
        decimal AvailableQuant;
        return AvailableQuant = Decimal.Parse(AvailableQuantity[0]);
    }

    //method used to retrieve unit price information
    [WebMethod]
    public string[] GetUnitPrice(string ProdId)
    {
        SqlCommand cmd = new SqlCommand("SELECT
QtyBreak1,QtyBreak2,QtyBreak3,UnitPrice1,UnitPrice2,UnitPrice3" + "
FROM SupplierPriceDistribution" + " WHERE ProductId = @ProdId", con);
        cmd.Parameters.AddWithValue("@ProdId", ProdId);
        string[] Data;
        return Data = SelectMethod(cmd);
    }

    //insert data into Supplier Price Distribution
    [WebMethod]
    public void InsertUnitPriceInfo(string ProdId, string QtyBreak1,
string QtyBreak2, string QtyBreak3, decimal UnitPrice1, decimal
UnitPrice2, decimal UnitPrice3)
    {
        SqlCommand cmd = new SqlCommand("INSERT INTO
SupplierPriceDistribution
(ProductId,QtyBreak1,QtyBreak2,QtyBreak3,UnitPrice1, UnitPrice2,
UnitPrice3)" + "
VALUES(@ProdId,@QtyBreak1,@QtyBreak2,@QtyBreak3,@UnitPrice1,@UnitPric
e2,@UnitPrice3)", con1);
        cmd.Parameters.AddWithValue("@ProdId", ProdId);
        cmd.Parameters.AddWithValue("@QtyBreak1", QtyBreak1);
        cmd.Parameters.AddWithValue("@QtyBreak2", QtyBreak2);
        cmd.Parameters.AddWithValue("@QtyBreak3", QtyBreak3);
        cmd.Parameters.AddWithValue("@UnitPrice1", UnitPrice1);
        cmd.Parameters.AddWithValue("@UnitPrice2", UnitPrice2);
        cmd.Parameters.AddWithValue("@UnitPrice3", UnitPrice3);

        UpdateInsertDelete(cmd);
    }


    //used to retreive the current capacity
    [WebMethod]
    public string[] currentProductionCapacity(string ProdId)
    {
        SqlCommand cmd = new SqlCommand("SELECT
ManufacturingCapacity" + " FROM SupplierShopFloorData" + " WHERE
ProductId = @ProdId", con);
        cmd.Parameters.AddWithValue("@ProdId", ProdId);
        string[] Data;
```

```
 return Data = SelectMethod(cmd);

    }

    [WebMethod]
    public void InsertShopFloorData(string MachineId, string ProdId,
decimal CPLinedUp, decimal MfrCapacity)
    {
        SqlCommand cmd = new SqlCommand("INSERT INTO
SupplierShopFloorData
(MachineId,ProductId,CurrentProductionLinedUp,ManufacturingCapacity)"
+ " VALUES (@MachineId, @ProdId, @CPLinedUp, @MfrCapacity", con1);
        cmd.Parameters.AddWithValue("@MachineId", MachineId);
        cmd.Parameters.AddWithValue("@ProdId", ProdId);
        cmd.Parameters.AddWithValue("@CPLinedUp", CPLinedUp);
        cmd.Parameters.AddWithValue("@MfrCapacity", MfrCapacity);

        UpdateInsertDelete(cmd);
    }

    //this gives the current production linedup
    [WebMethod]
    public string[] currentProduction(string ProdId)
    {
        SqlCommand cmd = new SqlCommand("SELECT
CurrentProductionLinedup" + " FROM SupplierShopFloorData" + " WHERE
ProductId = @ProdId", con);
        cmd.Parameters.AddWithValue("@ProdId", ProdId);

        string[] Data;

        return Data = SelectMethod(cmd);
    }

    //method to update offered price data in SupplierRFQData Table
    [WebMethod]
    public void UpdateOfferedPriceData(string rfqid, decimal
ModifiedunitPrice, DateTime earliestshipdate, string priceupdate)
    {
        SqlCommand cmd = new SqlCommand("UPDATE SupplierRFQData" + "
SET UnitPriceOffered = @ModifiedunitPrice, PromisedDeliverydate =
@earliestshipdate, PriceUpdateStatus = @status" + " WHERE
RFQDocumentID = @rfqid", con1);
        cmd.Parameters.AddWithValue("@ModifiedunitPrice",
ModifiedunitPrice);
        cmd.Parameters.AddWithValue("@rfqid", rfqid);
        cmd.Parameters.AddWithValue("@earliestshipdate",
earliestshipdate);
        cmd.Parameters.AddWithValue("@status", priceupdate);

        UpdateInsertDelete(cmd);
    }

    //method to retreive information related to MFRId and ProductId
    [WebMethod]
```

```
public string[] RetreiveMfrIdAndProductId(string RFQId)
    {
        string[] RFQData1;
        SqlCommand cmd = new SqlCommand("SELECT MfrId, ProductId,
Quantity, UnitPriceOffered, PromisedDeliveryDate,
RequestedDeliveryDate" + " FROM SupplierRFQData" + " WHERE
RFQDocumentID = @RFQId", con);
        cmd.Parameters.AddWithValue("@RFQId", RFQId);

        return RFQData1 = SelectMethod(cmd);

    }
    //method to retreive information related to Quantity and
UnitPriceOffered
    [WebMethod]
    public string[] RetreiveQuantityandUnitPO(string RFQId)
    {
        string[] RFQData2;
        SqlCommand cmd = new SqlCommand("SELECT Quantity,
UnitPriceOffered" + " FROM SupplierRFQData" + " WHERE RFQDocumentID =
@RFQId", con);

        cmd.Parameters.AddWithValue("@RFQId", RFQId);

        return RFQData2 = SelectMethod(cmd);
    }

    //method to retreive information related to Required delivery
date
    [WebMethod]
    public DateTime RetreiveReqDD(string RFQId)
    {
        string[] RFQData3;
        DateTime RFQReqdd;
        SqlCommand cmd = new SqlCommand("SELECT
RequestedDeliveryDate" + " FROM SupplierRFQData" + " WHERE
RFQDocumentID = @RFQId", con);
        cmd.Parameters.AddWithValue("@RFQId", RFQId);

        RFQData3 = SelectMethod(cmd);
        return RFQReqdd = DateTime.Parse(RFQData3[0]);
    }
    //method to update Purchase Order Data
    [WebMethod]
    public void UpdatePPOData(string ppoid, string mfrid, decimal
quant, decimal unitprice, string prodid, DateTime
earliestshipdatetime, DateTime requestedshipdatetime, string
rfqid,string status)
    {
        SqlCommand cmd = new SqlCommand("INSERT INTO
SupplierPurchaseOrderData
(PODocumentId,CustomerPartyId,UnitPrice,NumberOfItemsOrdered,Prodid,E
arliestShipDateTime,RequestedShipDateTime, Status, RfqId) VALUES " +
"(" + "@ppoid, @mfrid, @unitprice, @quant, @prodid,
```

```
    @earliestshipdatetime, @requestedshipdatetime, @status, @rfqid" +
")", con1);cmd.Parameters.AddWithValue("@ppoid", ppoid);
        cmd.Parameters.AddWithValue("@mfrid", mfrid);

//cmd.Parameters.AddWithValue("@customerpartyid",customerpartyid);
        //cmd.Parameters.AddWithValue("@totalamount", totalamount);
        cmd.Parameters.AddWithValue("@earliestshipdatetime",
earliestshipdatetime.ToShortDateString());
        cmd.Parameters.AddWithValue("@requestedshipdatetime",
requestedshipdatetime.ToShortDateString());
        cmd.Parameters.AddWithValue("@quant", quant);
        cmd.Parameters.AddWithValue("@prodid", prodid);
        cmd.Parameters.AddWithValue("@status", status);
        cmd.Parameters.AddWithValue("@rfqid", rfqid);
        cmd.Parameters.AddWithValue("@unitprice",unitprice);

        UpdateInsertDelete(cmd);

    }

    //Retreive PPOData to check if any changes are made when Process
Purchase Order is Submitted
    [WebMethod]
    public string[] RetreivePPOData(string purchaseOrderId)
    {
        string[] PPOData;

        SqlCommand cmd = new SqlCommand("SELECT EarliestShipDateTime,
NumberOfItemsOrdered, UnitPrice, ProdId" + " FROM
SupplierPurchaseOrderData" + " WHERE PODocumentId =
@purchaseOrderId", con);
        cmd.Parameters.AddWithValue("@purchaseOrderId",
purchaseOrderId);

        return PPOData = SelectMethod(cmd);
    }
    [WebMethod]
    public void UpdatePPOData1(string ppoid, string status)
    {
        SqlCommand cmd = new SqlCommand("UPDATE
SupplierPurchaseOrderData" + " SET Status = @status" + " WHERE
PODocumentId = @ppoid", con1);
        cmd.Parameters.AddWithValue("@status", status);
        cmd.Parameters.AddWithValue("@ppoid", ppoid);

        UpdateInsertDelete(cmd);
    }

    [WebMethod]
    public void InsertDataSuppInv(string ProdId, decimal availquant)
    {
        SqlCommand cmd = new SqlCommand("INSERT INTO
InventoryOfSupplier" + " VALUES (@ProdId,@availquant)", con1);
        cmd.Parameters.AddWithValue("@ProdId", ProdId);
        cmd.Parameters.AddWithValue("@availquant", availquant);
```

```
UpdateInsertDelete(cmd);
    }

    //method to update data in SupplierOrderedProductData, after
purchase order has been placed.
    [WebMethod]
    public void InsertintoOrderedProductData(string prodid,string
podocumentId, decimal unitprice, decimal quantityordered, decimal
quantityshipped)
    {
        SqlCommand cmd = new SqlCommand("INSERT INTO
SupplierOrderedProductData(ProductId,PODocumentID,UnitPrice,QtyOrdere
d,QtyShipped)" + "
VALUES(@prodid,@podocumentid,@unitprice,@quantityordered,@quantityshi
pped)", con1);
        cmd.Parameters.AddWithValue("@prodid", prodid);
        cmd.Parameters.AddWithValue("@podocumentId", podocumentId);
        cmd.Parameters.AddWithValue("@unitprice", unitprice);
        cmd.Parameters.AddWithValue("@quantityordered",
quantityordered);
        cmd.Parameters.AddWithValue("@quantityshipped",
quantityshipped);

        UpdateInsertDelete(cmd);
    }

    //retreive status info
    [WebMethod]
    public string RetreiveStatusInfo(string mfrid)
    {
        string[] data;
        SqlCommand cmd = new SqlCommand("SELECT Status" + " FROM
SupplierManufacturer" + " WHERE MfrId = @mfrid", con);
        cmd.Parameters.AddWithValue("@mfrid", mfrid);

        data = SelectMethod(cmd);

        return data[0];
    }



    //method to write information to suppliermanufacturer table
    [WebMethod]
    public void InsertIntoSupplierManufacturer(string mfrid, string
address, string contactname, string contacttel, string email, string
preferredcarrierpartyid)
    {
        string status = "New";
        SqlCommand cmd = new SqlCommand("INSERT INTO
SupplierManufacturer(MfrId,Address1,MContactName,MEmail,PreferredCarr
ierPartyId, MContactTel, Status)" + "
VALUES(@mfrid,@address,@contactname,@email,@preferredcarrierpartyid,
@contacttel, @status)", con1);
```

```
cmd.Parameters.AddWithValue("@mfrid", mfrid);
            cmd.Parameters.AddWithValue("@address", address);
            cmd.Parameters.AddWithValue("@contactname", contactname);
            cmd.Parameters.AddWithValue("@contacttel", contacttel);
            cmd.Parameters.AddWithValue("@email", email);
            cmd.Parameters.AddWithValue("@preferredcarrierpartyid",
preferredcarrierpartyid);
            cmd.Parameters.AddWithValue("@status", status);

            UpdateInsertDelete(cmd);
        }

        //method to retrieve mfrid
        [WebMethod]
        public string RetreiveMfrid(string podid)
        {
            string[] Data;
            SqlCommand cmd = new SqlCommand("Select CustomerPartyId" + "
FROM SupplierPurchaseOrderData" + " WHERE PODocumentId = @podid",
con);

            cmd.Parameters.AddWithValue("@podid", podid);

            Data = SelectMethod(cmd);
            return Data[0];
        }

        //method to update the status
        [WebMethod]
        public void UpdateMfrStatus(string mfrid, string mfrstatus)
        {
            SqlCommand cmd = new SqlCommand("UPDATE SupplierManufacturer"
+ " SET Status = @mfrstatus" + " WHERE MfrId = @mfrid", con1);
            cmd.Parameters.AddWithValue("@mfrstatus", mfrstatus);
            cmd.Parameters.AddWithValue("@mfrid", mfrid);

            UpdateInsertDelete(cmd);
        }

        //method to update ppodata on request for change of ppo.
        [WebMethod]
        public void UpdatePPOOnChangeRequest(string ppoid, decimal
reqquant, decimal unitprice, DateTime reqdd)
        {
            SqlCommand cmd = new SqlCommand("UPDATE
SupplierPurchaseOrderData" + " SET NumberOfItemsOrdered = @reqquant,
UnitPrice = @unitprice, RequestedShipDateTime = @reqdd" + " WHERE
PODocumentId = @ppoid", con1);

            cmd.Parameters.AddWithValue("@ppoid", ppoid);
            cmd.Parameters.AddWithValue("@reqquant",reqquant);
            cmd.Parameters.AddWithValue("@unitprice",unitprice);
            cmd.Parameters.AddWithValue("@reqdd",reqdd);
            UpdateInsertDelete(cmd);

        }
```

```
    //method to retreive productid from SupplierPurchaseOrderData
table
    [WebMethod]
    public string[] RetreiveProdIDfromSPOD(string poid)
    {
        string[] data;

        SqlCommand cmd = new SqlCommand("SELECT ProdId, UnitPrice,
EarliestShipDateTime, NumberOfItemsOrdered" + " FROM
SupplierPurchaseOrderData" + " WHERE PODocumentId = @poid", con);
        cmd.Parameters.AddWithValue("@poid", poid);

        data = SelectMethod(cmd);

        return data;
    }

    //method to retreive PromisedShipDateTime and status info from
purchase order data
    [WebMethod]
    public string[] RetreivePromisedDDandStatus(string ppoid)
    {
        string[] data;
        SqlCommand cmd = new SqlCommand("SELECT EarliestShipDateTime,
Status, UnitPrice" + " FROM SupplierPurchaseOrderData" + " WHERE
PODocumentId = @ppoid", con);
        cmd.Parameters.AddWithValue("@ppoid", ppoid);

        data = SelectMethod(cmd);
        return data;

    }

    protected string[] SelectMethod(SqlCommand cmd)
    {

        try
        {
            Application.Lock();
            string[] returnValue;
            ///using list to store the information
            List<string> lst = new List<string>();

            using (con)
            {
                try
                {
                    ///open the connection
                    con.Open();
                    ///retrive data from the datasource
                    SqlDataReader reader = cmd.ExecuteReader();
                    while (reader.Read())
                    {
                        for (int i = 0; i < reader.FieldCount; i++)
                        {
```

```
lst.Add(reader.GetValue(i).ToString());
                        }
                    }
                }
                finally
                {
                    ///if no value is returned then add a null value
to the list.
                    if (lst.Count == 0)
                    {
                        lst.Add(null);

                    }
                    ///close the connection
                    con.Close();
                }
            }

            returnValue = lst.ToArray();
            return returnValue;
        }
        finally
        {
            Application.UnLock();


        }
    }

    //method to update data in the database

    protected void UpdateInsertDelete(SqlCommand cmd)
    {
        try
        {
            Application.Lock();

            using (con1)
            {
                con1.Open();
                cmd.ExecuteNonQuery();
                con1.Close();
            }
        }
        finally
        {
            Application.UnLock();
        }
    }

}
```

Supplier KnowledgeSource web service code:

```
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;


[WebService(Namespace = "http://rit.edu/~asi")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    decimal unitPrice;
    public Service()
    {

        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    public string HelloWorld()
    {
        return "Hello World";
    }

    //decide unit price
    //receives three input parameters, required quantity, unitprice
distribution and available quantity
    [WebMethod]
    public decimal DecideUnitPrice(decimal AvailableQuant, decimal
quantity, string[] Data, bool rfq)
    {
        if (quantity <= decimal.Parse(Data[0]))
        {
            unitPrice = decimal.Parse(Data[3]);
        }
        else if (quantity <= decimal.Parse(Data[1]))
        {
            unitPrice = decimal.Parse(Data[4]);
        }
        else
        {
            unitPrice = decimal.Parse(Data[5]);
        }
        if ((AvailableQuant >= quantity) && (rfq == true))
        {
            return unitPrice;
        }
        else if ((AvailableQuant <= quantity) && (rfq == false))
        {
            return unitPrice = unitPrice + (unitPrice *
decimal.Parse("0.15"));
        }
        else
        {
```

```
            return unitPrice = unitPrice - (unitPrice *
decimal.Parse("0.15"));
        }
    }

    //webmethod to determine the earliest possible delivery date
    [WebMethod]
    public DateTime EarliestPossibleDeliveryDate(decimal linedupprod,
decimal sum, decimal quantity, int carrierdays)
    {
        DateTime earliestreturndate, earliestreturndateinter2;
        decimal earliestreturndateinter1;

        earliestreturndateinter2 = DateTime.Now;
        earliestreturndateinter1 =  ((linedupprod / sum) + (quantity
/ sum) + carrierdays + 2);
        //earliestreturndateinter1 = int.Parse((linedupprod /
sum).ToString()) + int.Parse((quantity / sum).ToString()) +
carrierdays + 2;
       // earliestreturndate =
        earliestreturndate =
((earliestreturndateinter2).AddDays(double.Parse((decimal.Round(earli
estreturndateinter1)).ToString())));

        return earliestreturndate;
    }

    //method to decide new price to change ppo request
    //there is a request for change ==> i should check the
manufacturer status ,
    //4 levels of status. 1 - VGood - offer 10% discount on price
    //                    2 - Good  - offer 5%  discount on price
    //                    3 - Poor  - No discount is offered
    //                    4 - New   - 5% discount is offered.
    [WebMethod]
    public decimal DecideUpForChangeRequest(string status, decimal
oldprice)
    {
        decimal new_price;
        if (status == "VGood")
        {
            new_price = oldprice - oldprice *
Decimal.Parse((0.1).ToString());
        }
        else
        {
            new_price = oldprice - oldprice *
Decimal.Parse((0.05).ToString());
        }

        return new_price;

    }

    //Method to decide the penalty to be imposed on cancellation
```

```
[WebMethod]
    public string PenaltyCalculation(DateTime promiseddd, decimal
unitprice)
    {
        string[] data = new string[2];
        DateTime present = System.DateTime.Now;

        TimeSpan diff = promiseddd.Subtract(present);
        int days= diff.Days;

        if (days < 7)
        {
            data[1] = (0.0).ToString();
            data[0] = " You Cannot Cancel Your Order" + data[1];

        }
        else if (days < 14)
        {
            //30%penalty
            data[1] = (unitprice*decimal.Parse("0.3")).ToString();
            data[0] = "You have been charged a penalty of $" +
data[1] + " per unit";
        }
        else
        {
            //no penalty
            data[0]="Your PurchaseOrder has been successfully
cancelled";
        }



        return data[0];
    }


}
```