Multi-agent system for negotiation in virtual supply chain management

You, Qi

2006

You, Q. (2006). Multi-agent system for negotiation in virtual supply chain management. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/36113>

<https://doi.org/10.32657/10356/36113>

9457244

A Multi-agent System for Negotiation in Virtual Supply Chain Management

You Qi

School of Mechanical and Aerospace Engineering

A thesis submitted to
Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Master of Engineering

2006

ABSTRACT

A virtual supply chain (VSC) may emerge from the negotiation process among possible supply chain partners with an attempt to achieve the common goals. Successful virtual supply chain management (VSCM) thus highlights the cross-functional integration of key business processes within a company and across the network of companies that comprise the VSC. In the paradigm of multi-agent systems (MASs), a VSC can be viewed as a set of intelligent agents and the interaction among them.

In an MAS, functional agents negotiate cooperatively to make decisions so as to fulfill the common goals and own interests. Therefore, an MAS can provide an effective means to manage a VSC through building agents as supply chain partners.

This thesis proposes a negotiation-based MAS (NBMAS) for VSCM. The efficient negotiation processes among supply chain partners (agents) are modeled. A prototype of NBMAS for VSCM, which uses Java Development Agent (JADE) software, has been developed and reported in the case study. The results show the effectiveness and efficiency of the proposed model for managing supply chains.

ACKNOWLEDGEMENT

I am deeply grateful to my supervisor, Dr. Roger Jiao for his guidance, encouragement and inspiration throughout the progress of this thesis.

I appreciate all the help and inspiration that I got from my fellows, Ms. Zhang Lianfeng, Mr. He Qi, Mr. You Xiao, and other friends in Nanyang Technological University, Singapore. Their cooperation and patience helped me focus on my thesis without other troubles.

Special thanks are due to Mr. Cui Yong for his moral support and encouragement. Last but not least, I would like to express my deepest love and appreciation to my dear family. This thesis would never be possible without their care, love, understanding, and endless support.

Table of Contents

Abstract..... I

Acknowledgement..... II

Table of Contents..... III

List of Figures..... V

List of Tables VI

Chapter 1 Introduction..... 1

 1.1 Background..... 1

 1.2 Objective..... 3

 1.3 Scope 3

 1.4 Dissertation Organization 3

Chapter 2 Literature Review 6

 2.1 Virtual Supply Chain Management..... 6

 2.2 Intelligent Supply Chain Management..... 8

 2.3 Intelligent Agent System 10

 2.3.1 Agent classification 12

 2.3.2 Multi-agent system 13

 2.3.3 Negotiation technique..... 15

 2.4 Agent-based Supply Chain Management 17

 2.4.1 Agent-based SCM implementation 17

 2.4.2 Mobile agent-based SCM 19

Chapter 3 Conceptual MAS for VSCM..... 20

 3.1 VSC Lifecycle 20

 3.2 Conceptual MAS Architecture 22

 3.3 Negotiation Mechanism..... 26

 3.4 Java Agent Development Framework 29

 3.4.1 JADE platform..... 31

 3.4.2 Agent communication 32

Chapter 4 Proposed Negotiation-based MAS..... 36

 4.1 Negotiation Process 36

 4.2 System Architecture..... 37

Table of Contents

4.2.1	Databases	39
4.2.2	Negotiation and selection rules	40
Chapter 5	Case Study	42
5.1	Company Background	42
5.2	Business Process.....	43
5.3	Implementation.....	47
5.4	Execution.....	48
5.5	Evaluation.....	51
Chapter 6	Conclusions and Recommendations	53
6.1	Conclusions	53
6.2	Contributions	53
6.3	Limitations and Future Work.....	54
Reference		56
Appendix		61

List of Figures

Figure 1.1 Organization of the thesis.....	5
Figure 3.1 VSC lifecycle	22
Figure 3.2 Architecture of an MAS	23
Figure 3.3 Negotiation mechanism.....	28
Figure 3.4 Architecture of a FIPA agent platform	31
Figure 3.5 JADE agent platform distributed over several containers	32
Figure 3.6 Communication between agents	35
Figure 4.1 Negotiation process.....	36
Figure 4.2 System architecture	38
Figure 5.1 Motor description.....	43
Figure 5.2 Purchase process	43
Figure 5.3 Customer order information.....	48
Figure 5.4 Detailed information of component	48
Figure 5.5 The starting of JADE platform.....	49
Figure 5.6 JADE GUI for agents.....	49
Figure 5.7 ACL message GUI	50
Figure 5.8 The result.....	50
Figure 5.9 System outcome of the process.....	51
Figure 5.10 Feasible suppliers selection.....	52

List of Tables

Table 3.1 JADE technical details.....30

Table 5.1 Purchase requisition44

Table 5.2 Supplier1 information.....45

Table 5.3 Supplier2 information.....45

Table 5.4 Supplier3 information.....45

Table 5.5 Supplier4 information.....46

Table 5.6 Supplier5 information.....46

Table 5.7 Supplier6 information.....46

Table 5.8 Supplier7 information.....46

Chapter 1 Introduction

1.1 Background

The concept of supply chain (SC) is about managing coordinated information and material flows, plant operations and logistics (Lee and Billington, 1993). The high-level coordination within various supply chain partners is achieved through the real-time negotiation among partners. Increased coordination can lead to reduction in lead times and costs, alignment of interdependent decision-making processes and improvement in the overall performance of each partner, as well as the supply chain network as a whole (Chandra, 1997).

To survive in current global manufacturing environment, companies are forced to work efficiently with multiple partners. Thus, rapid and “virtual” symbiosis has become the linchpin of the new supply chain management strategies for potential partners to work together to attain the biggest profits. To support the migration process from internal only to extended supply chains, a great number of software and hardware tools have been developed, allowing collaboration through the entire electronic SC connection. Consequently, the concept of virtual factory or virtual supply chain (VSC) has been emerging.

Virtual supply chain management (VSCM) puts emphasis on customer-driven and partnership synchronization. However, successful VSCM requires efficient interacts among cross-functional firms that comprise the supply chains. One of the main issues is how to find and coordinate the potential partners for the common competing

Introduction

objectives and own interests. All participants share information during the course of communication with others and provide decision-making capabilities for achieving their shared goals. Intelligent agent technology offers such negotiation platforms and a potential to overcome many limitations of other technologies that are adopted to manage supply chain.

In recent years, multi-agent software for managing the SCs has received much attention from researchers. Such systems address supply chain issues at both the tactical and operational levels. They view a SC as a set of intelligent agents. Each agent is responsible for one or more activities in the chain and interacts with other agents in planning and executing their responsibilities. An agent is an autonomous, goal-oriented software process that operates asynchronously, communicates and coordinates with other agents as required (Fox et al., 2000). Multi-agent system (MAS) thus facilitates the coordination and integration among supply chain partners, e.g., buyers and suppliers.

When an order comes from the end customer, a virtual supply chain may emerge through negotiation processes. It is very complex to select the most qualified ones among multiple suppliers with their own interests. A VSC is temporary and loose, so partners may join or leave the chain according to the different requirements of different customer orders. To address these problems, a negotiation-based MAS (NBMAS) is proposed for VSCM. In this system, there is no preset relationship between functional agents with the specific negotiation rules. These computational

Introduction

and self-acting agents facilitate negotiation and selection to coordinate and complete the purchase process effectively.

1.2 Objective

The linchpin of successful VSCM with respect to select proper supply chain partners in order to obtain the most added values lies in negotiation and coordination among partners in a VSC. Attempting to facilitate supply chain partners to interact efficiently with one another, this thesis proposes a model of NBMAS. The ultimate goal is to assist companies in selecting the best-fit partners from their large supplier base in response to customer demand changes.

1.3 Scope

In this thesis, a proposed NBMAS is designed for VSCM. Java Development Agent (JADE) software is adopted as the agent platform to develop the prototype which focuses on purchase process in distributed and internet-enabled environment.

1.4 Dissertation Organization

This thesis is structured into 5 Chapters, as shown in Figure 1.1.

Chapter 2 introduces the concepts of VSC, the roles of MASs and intelligent agents as well as their properties and the characteristics of the agent-based virtual supply chains. Chapter 3 presents a conceptual model of MAS for VSCM. Also introduced are JADE and the negotiation mechanism of the conceptual model.

Introduction

In chapter 4, the proposed model of negotiation-based MAS for VSCM is discussed in detail. The prototype of the proposed system has been developed and explained in Chapter 5. Some findings are drawn in Chapter 6. The avenues of possible future work are also given in this chapter.

Figure 1.1 Organization of the thesis

Chapter 2 Literature Review

2.1 Virtual Supply Chain Management

Recent advances in information technology (IT) have brought dramatic changes to supply chain management (SCM) (Ngai and Cheung, 2004). The traditional methods of inventory control warehousing and scheduling of deliveries for a company have to be completely amended in order to deal with the demands of customers, which have been highly complex with the advent of e-commerce and trade globalization. The emerging business strategy is set up towards maximizing the benefits and optimally sharing the risks and resources through collaborations. All the partners should be coordinated and managed as a single unit. This leads to establish virtual supply chain management (VSCM).

Virtual supply chain (VSC) is a new supply chain management venture. The company provides a managed service which is designed to help medium to large-sized corporations reduce their costs and increased business efficiencies through closer B2B integration and greater transparency in their supply chains. VSC recognizes that while many of the world-class large companies have pioneered closer application integration with their trading partners historically through EDI and now via internet-based trading communities and integrated systems.

Although the academia has reached a consensus on the definition of a supply chain which is a network consisting of supplier, manufacturer, distributor, logistic and

Literature Review

retailer, where information flows, material flows and financial flows are moving from upstream to downstream or both directions, numerous definitions of VSCs reflect different views on virtuality:

- ♦ A VSC is an organizational network which is structured and managed in such a way that it is operated via customers and other external stakeholders as an identifiable and complete organization (Aken, 1998). The underlying idea of this definition is that a VSC is identified as one organization, but in fact consists of many different organizations. This corresponds with the view of virtuality of being unreal, looking real.
- ♦ A VSC is a temporary network of independent companies linked by information technology to share skills, costs and access to one another's market. The companies unite quickly to exploit a specific opportunity and will disperse afterward (Byrne, 1993). This definition encloses three views on virtuality, namely, immaterial, supported by ICT and potentially present and also existing, but changing.
- ♦ The essence of VSC is the meta management of goal-oriented activity in a way that is independent of the means for its realization. Meta management is the management of a virtually organized activity. Such a virtually organized activity contains a set of requirements, a set of satisfiers and a procedure to map the satisfiers on the requirements (Mowshowitz, 1994). This definition relates to a way that a VSC is composed of. This composition depends on the requirements a VSC has and how other organizations can satisfy these requirements.

Literature Review

While all these definitions address their particular characteristics, there are also some common aspects in these definitions. They include:

- 1) A temporary, cooperative network that is formed by independent, autonomous companies to exploit a particular market opportunity; and
- 2) Under the support of information and communication technology.

A VSC can be treated as a chain of virtual individuals or organizational partners with the capabilities of communicating via communication platforms at any time and place. In such a system, all nodes of the chain should be implemented in a way that has the below abilities:

- ♦ Connect to other nodes via a communication technology. The communication platform can be of any kind;
- ♦ Make decision based on the information acquired; and
- ♦ Detect probable errors and problems.

2.2 Intelligent Supply Chain Management

Increasingly the management of relationships across a supply chain is being referred to as supply chain management (SCM). SCM represents a critical competency in today's fast-paced, global business environment. The many solutions of SCM have lead to a number of effective practices, e.g., just-in-time deliveries, supplier inventory management. It is employed to improve the competitiveness and efficiency of enterprises around the world (Nissen, 2001).

Literature Review

Strictly speaking, the supply chain is not a chain of business with one-to-one, business-to-business relationships, but a network of business and relationship. Therefore, management of multiple partners is becoming critical to an enterprise's success. There are two important keys to achieve that success: a thorough understanding of the business processes and quality practices at these suppliers, and the synchronization of material flow and information flow across the chain (Umeda and Jones, 1997).

Because of a large number of transactions that occur within a single supply chain, controlling and handling the material and information flows is complex. A major challenge facing companies with integrated information systems is how to process and utilize information available to users within the chain. To deal with this situation, companies are beginning to introduce new types of "intelligent" decision support systems. Such systems offer a three-tiered vehicle that allows (Handfield and Nichols, 1999):

- Better planning and decision making via intelligent decision support tools;
- Network systems with intelligent communications support; and
- Enterprise systems that offer intelligent operations response.

Intelligent decision support systems allow users to model the different scenarios and variations in a supply chain. Such models consider many factors, e.g., costs, constraints, business objectives and other variables to make the best decisions based on simulated trials (Handfield and Nichols, 1999).

Literature Review

Intelligent supply chain management (ISCM) can be implemented in two ways: one is decision support system (DSS), and the other is Agent based systems and multi-agent based systems (MASs) (Tarokh and Tayebi, 2005). In the method of DSS based SCM, each node in a SC is a DSS, and is managed separately. There is a knowledge base server (KBS) including system knowledge base and data mining tools, which is responsible for gathering and mining information from all parts of the SC and supports the whole SC with the resulted knowledge. Each node will be served by information and knowledge saved in the KBS. It can query any required information from the KBS. Such a system is good on the point of its integration but suffered from high data processing load on the server. In the case of any failure on server, everything would go wrong.

The other method is to use agent-based SCM, which is the most under research method nowadays, and is moving from research papers and laboratories to real implementation. In this method every node of a chain is a software intelligent agent. Agent-based software engineering is invented to facilitate information and service exchange with other programs and thereby solve problems that cannot be solved alone (Genesereth and Ketchpel, 1994).

2.3 Intelligent Agent System

Computer software and hardware development has lead to the appearance of non-human software agencies. A software agent can be treated as an entity with its own goals and is capable of actions endowed with domain knowledge and situated in

Literature Review

an environment. An MAS is suitable for the domains that involve interactions among different people or organizations with different (possibly conflicting) goals and proprietary information (Stone and Veloso, 1997).

In the computational world, roles of individual entities in a SC are implemented as distinct agents. Correspondingly, a supply chain management system (SCMS) can be transformed to an MAS, in which functional agents cooperate with one another in order to fulfill system functionalities. Most of the previous research work in this field sets MAS in a closed environment, that is, the system consists of a fixed number of entities/components and they have a common target (Muller, 1996).

The coordination of chain components is a hierarchical scheduling problem (Kjenstad, 1998). However, this setting can not reflect accurately the real situation in which a SC sits due to two reasons. First, every company in the SC has its own interests and goals even though companies may also have intentions to deal with one another. The existence of self-interest makes it difficult to model the agent cooperation as a pure scheduling problem. Second, in a real business environment there are no obligations for companies to remain with a SC for a certain time period. Companies may join or leave the chain according to their own judgment. In other words, functional agents have to cooperate in a relatively dynamic way and in an open environment. To address this problem, negotiation technology should be adopted in MAS framework for SCM. There is no preset relationship between functional agents. A VSC may emerge through negotiation processes. The components of the chain may change according to the external situation.

2.3.1 Agent classification

An agent (also called an intelligent agent) is a program that gathers information or performs other services without users' immediate presence (Nwana et al., 1998). They can be defined in different ways depending on the way they are implemented and the tasks they perform. Wooldridge and Jennings (1995) suggest that any computer system (software or hardware) should have the following properties to be termed as an agent (Wooldridge and Jennings, 1995):

- **Autonomy:** it should have some control over its actions and should work without human intervention;
- **Social ability:** it should be able to communicate with other agents and/or with human operators;
- **Reactivity:** it should be able to react to changes in its environment; and
- **Pro-activeness:** it should also be able to take initiative based on pre-specified goals.

According to the architecture and platform environment, agents can be classified into two types as follows:

- 1) Stationary agents are agents, free of modification for a certain period and are designed to work in a specific environment with reliable communications, enough bandwidth and with limited transactions i.e., sending/receiving messages. However, literature review suggests that stationary agent isn't a good answer to intelligence in SCM due to the classification of SCM platforms, e.g., variable behavior and limited bandwidth.
- 2) Mobile agents are mobile, flexible, autonomous, dynamic and efficient. They

can move around different networks. When encapsulated within a task, a mobile agent can be dispatched to a remote host by the original host. After executing and accomplishing its tasks at the remote host, it can bring the results back by returning to the original host or send them through a message. The mobile agent approach is also suitable for deploying parallel processes over distributed sites on the Internet. The tasks can be decomposed and encapsulated into multiple mobile agents. Every mobile agent can run independently to accomplish its task. Thus, a set of mobile agents can run in parallel on distributed hosts so that the whole task can be completed in a shorter time (Sahingoz and Erdogan, 2003).

2.3.2 Multi-agent system

An MAS is viewed as a loosely coupled network of problem solvers that interact to solve problems. These problems are beyond the individual capabilities or knowledge of each problem solving (Durfee et al., 1989). An MAS consists of a group of agents that can take specific roles within an organizational structure. Different types of agents may represent different objects, with different authorities and capabilities, and perform different functions or tasks. They can support a group of distributed and independent entities working cooperatively in a dynamic and flexible manner.

In order for MAS to solve common problems coherently, the agents must communicate amongst themselves, coordinate their activities. Coordination and communication are central to MAS, for without it, any benefits of interaction vanish

Literature Review

and the group of agents quickly degenerates into a collection of individuals with a chaotic behaviour. The characteristics of MASs are as follows:

- ♦ Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint;
- ♦ There is no system global control;
- ♦ Data are decentralized; and
- ♦ Computation is asynchronous.

Applications of MASs in manufacturing and supply chain management are not new. In intelligent manufacturing, agents have been used in following functional areas: manufacturing control, configuration design, collaborative design, agile manufacturing scheduling and manufacturing operations control. Shaw and Whinston (1983) perhaps provide the earliest idea of the agents with the capabilities of subcontracting and bidding for manufacturing scheduling and factory control. Shaw (1998) follows up with a framework for networked decision making in cellular manufacturing systems. Pan and Tenenbaum (1991) develop an intelligent agent framework for integrating people and computer systems in a large, geographically dispersed manufacturing enterprise. Lin and Solebreg (1992) develop an integrated shop floor control model to achieve real-time agents coordinate. Sikora and Shaw (1998) provide a multi-agent framework for the coordination and integration of information systems. Gilman (1997) integrates the design and manufacturing in a virtual enterprise using enterprise rules, intelligent agents, steps and work flows.

2.3.3 Negotiation technique

It appears that almost every form of human interaction involves some degree of explicit or implicit negotiation (Muller, 1996). Negotiation techniques are based upon realistic assumptions. They provide a more suitable basis for automation and can therefore be used in a wider variety of application domains.

Sycara (1989) identifies four issues that underpin agent negotiation techniques as follows:

- 1) Ensuring network coherence;
- 2) Facilitating task and resource allocation among agents;
- 3) Recognizing and resolving disparities in agents' goals and viewpoints; and
- 4) Determining organizational structures.

Agents make their negotiation decision according to some kind of utilities they will gain, which are affected by the utilities and decisions of other agents. Three distinct domains where negotiation is applicable have been identified (Rosenschein, 1986):

- 1) Task-oriented domain: agents negotiate to allocate their tasks that will benefit everyone;
- 2) State-oriented domain: agents negotiate to find actions that change the state of the "world" and serve their goals; and
- 3) Worth-oriented domain: agents negotiate to gain maximum utilities for reaching certain states.

From an intra-agent perspective, negotiation is a natural mechanism for agents to

Literature Review

resolve their differences by changing their viewpoints and finding mutually acceptable agreements. Most coordination schemes involve some sort of negotiation. Negotiation is used to modify agents' local decision-making procedures, and identify situations where certain agent interactions are possible. Modification and identification trigger the process of negotiation so that a common decision can be reached (Muller, 1996).

From a multi-agent system perspective, specific negotiation techniques are required to solve realistic problem. The common negotiation topics are tasks, plans, resources, intentions, preferences, or goals of agents. Agents can be either cooperative or competitive (self-interested) during negotiation. Complex structures are also allowed such that multiple agents may form a coalition or establish a separate agreement.

Agent communication languages (ACLs) are used during negotiation. ACLs are the standard formats for the exchange of messages. The DARPA-funded knowledge sharing effort (KSF) has defined two languages for agent communication: the knowledge query and manipulation language (KQML) and the knowledge interchange format (KIF) (labrou et al., 1999). KQML specifies an envelope format for messages. Each message has a performative and a number of parameters, such as the receiver of the message and the ontology used. KIF is a language for expressing message content. It facilitates the exchange of knowledge of particular domains. In 1995, the Foundation for Intelligent Physical Agent (FIPA) started to develop standards for multi-agent systems. The FIPA ACL is developed for specifying a standard message language by setting out the encoding, semantics and pragmatics of

the message. Thus, MAS can be established via communication mechanism and some development environment, e.g., RETSINA, JADE, MASCOT, and ZEUS.

2.4 Agent-based Supply Chain Management

Agent-based SCM is a generic and common way of implementing ISCM. In this method, SCM will be divided into a set of different activities, and each one will be managed with a special intelligent software agent (Wagner et al., 2002)

2.4.1 Agent-based SCM implementation

An agent-based SC is viewed as a set of intelligent agents (Lee and Whang, 1998). Each agent is responsible for one or more activities in the SC and each interact with other agents in the planning and execution of their responsibilities. An ISCM system should possess the following characteristics (Fox et al., 1993):

- ♦ Distributed: the functions of supply chain management are divided among a set of separate, synchronous software agents;
- ♦ Dynamic: each agent performs its functions asynchronously as required, as opposed to a batch or periodic mode;
- ♦ Intelligent: each agent is an “expert” in its function uses artificial intelligence and operations research problem solving methods;
- ♦ Integrated: each agent is aware of and can access the functional capabilities of other agents;
- ♦ Responsive: each agent is able to ask for information and/or a decision from

Literature Review

another agent - each agent is both a client and a server;

- ♦ Reactive: each agent is able to respond to events as they occur modifying its behavior as required, as opposed to responding in a pre-planned, rigid, batch approach;
- ♦ Cooperative: each agent can cooperate with other agents in finding a solution to a problem – that is, they do not act independently;
- ♦ Interactive: each agent may work with people to solve a problem;
- ♦ Anytime: no matter how much time is available, an agent is able to respond to a request, but the quality of the response is proportional to the time given to respond;
- ♦ Complete: the total functionality of the agents must span the range of functions required to manage the SC;
- ♦ Reconfigurable: the supply chain management system itself must be adaptable and must support the “relevant subset” of software agents. For example, if the user only wants to schedule a plant, he/she should not be required to use or have a logistics component;
- ♦ General: each agent must be adaptable to as broad a set of domains as possible;
- ♦ Adaptable: agents need to quickly adapt to the changing needs of the human organization. For example, adding a resource or changing inventory policy should be quick and easy for the user to do; and
- ♦ Backwards compatible: agents need to have a seamless upgrade path so that the release of new or changed features does not compromise existing integration or functionality.

2.4.2 Mobile agent-based SCM

Mobile agent-based SCM systems are expected to work completely electronically, and will be based on e-business transactions between mobile agents. Therefore, the platform for implementing such system is the Internet. Such system should be designed in such a way that it meets the following specific requirements of the distributed information systems for the industry:

- ♦ **Heterogeneity:** an Internet based SCM system integrates heterogeneous legacy systems which are heterogeneous and not designed to be interactive. They are usually implemented using different technologies, different languages and for different operating platforms;
- ♦ **Scalability:** an Internet based SCM should be able to integrate new applications into the distributed system without affecting the other components of the system; and
- ♦ **Reconfigurable:** the interactions between the local factories of the SC should be flexible and easy.

Chapter 3 Conceptual MAS for VSCM

This chapter presents a conceptual model of agent-based systems for VSCM. First, the lifecycle of a VSC is introduced. In section 3.2, the conceptual model of an agent-based system for VSCM is discussed in details. The negotiation mechanism of the conceptual model is described in section 3.3. The possible toolkit to implement such system, JADE is introduced in detail in section 3.4.

3.1 VSC Lifecycle

In a supply chain network, when a customer presents his order requirements, a dynamic VSC is created accordingly to fulfill the end customer order. The lifecycle of VSCM includes six phases as follows:

- 1) Initiation phase. In this phase, objectives, strategies, work flows, constraints, and duration are formed. Thus, this phase provides inputs to the definition of requirements and high-level business processes of the VSC.
- 2) Planning phase. This phase describes end customer requirements by orders or quotas. In doing so, these requirements are converted to product specifications, detailed structures, and bill of material (BOM). Cost structures and quality performance of the entire chain are also clarified in this phase.
- 3) Design Phase. In this phase, the necessary information and resources for implementing an identified VSC are outlined.
- 4) Formation Phase. VSC members are evaluated and selected based on the

Conceptual MAS for VSCM

identified technology, resources and capability requirements in the planning phase. Execution starts with each member receiving specific tasks from upstream entities (customers) and identifying and negotiating with its downstream entities (suppliers). The activities, for example, allocating resources and developing production plans and delivery plans, are performed.

- 5) Operation Phase. Process of monitoring and controlling process is carried out in this phase. While supplier chain partners perform their allocated work and provide materials to the upstream customers in the VSC, they cooperate and coordinate with one another to introduce rapidly new products to the market, reduce overall system costs, and improve customer service and responsibility.
- 6) Decommission Phase. It is the end of a VSC lifecycle. Temporary relationships among the participants within the VSC are terminated in this phase. The resources (personnel and hardware) will be reassigned to other VSCs.

Figure 3.1 captures the essentials of a VSC lifecycle. Each phase is represented by a solid box, while the constraints and activities formed in this phase are represented by a dashed box. The arrows connect phases and transmit information or objects between them. Arrows entering a box are inputs of activities occurred in the represented phase. Arrows leaving a box are the outputs of the phase.

Conceptual MAS for VSCM

Figure 3.1 VSC lifecycle

3.2 Conceptual MAS Architecture

The characteristics of the lifecycle of a VSC should be taken into account when designing an MAS system for VSCM. A conceptual MAS is presented in this section. Also introduced is negotiation mechanism.

Figure 3.2 shows the architecture of MAS for VSCM. The system is designed by introducing some functional agents, including logistics agent (LA), supplier agent (SA), control agent (CA), purchasing agent (PA) and etc. Dashed arrows represent information flow among agents, while solid arrows represent material flow among supply chain entities represented by agents. Agents inside a rectangle box belong to a same supply chain tier.

Conceptual MAS for VSCM

Figure 3.2 Architecture of an MAS

Each agent has its specific functionality. All the agents work together towards a common goal based on the Internet. The roles, functionalities of agents are introduced as follows:

1) Control agent

A control agent (CA) acts as a facilitator and coordinator in an MAS. It registers services offered and requested by individual agents and dynamically connects available services to requests. Agents send messages to the CA and publish information in CA's blackboard which is shared and accessible by all agents. For the unexpected events, for example, order alternation, new product introduction, material delay, the CA utilizes its reasoning engine to analyze the affection to the VSC, provide solution and modify corresponding agents to take relevant actions.

Conceptual MAS for VSCM

2) Retailer agent

A retailer agent (RA) is responsible for acquiring orders from customers, negotiating with customers about prices, due dates, etc., and handling customer requests for modifying or canceling respective orders. An RA is one of the agents participating in negotiation for creating supply chain plans.

3) Logistics agent

A logistics agent (LA) is responsible for transferring materials among plants, suppliers and distribution centers to achieve the best possible results in terms of the goals of the formed supply chain. It assigns and schedules transportation resources in order to satisfy interplant movement request. It is able to consider a variety of transportation assets and transportation routes in the construction of its schedules. The inputs to an LA are customer orders and deviations in factory schedules. The outputs are transportation plans for each plant, supplier, etc.

4) Warehouse agent

A warehouse agent (WA) models the material-handling department of an enterprise. It issues the required commodity to other agents and replenishes raw materials stock by procuring from other enterprises.

5) Purchasing agent

A purchasing agent (PA) models the purchase organization in an enterprise. It generates purchase orders, negotiates with suppliers and selects the optimal suppliers.

6) Supplier agent

A supplier agent (SA) represents the vendor company. After receiving a purchasing request from a PA, it takes part in the bidding and negotiation with other competitive suppliers.

An MAS can fulfill the following key functionalities:

- ♦ **Coordination.** Entities in a supply chain are independent organization with varied objectives and constraints. It is important to coordinate the behaviors of individual agents in a VSC to achieve a common goal. An MAS provides an adaptive coordination mechanism to support various VSCs existing in the supply chain network. CA of an MAS is able to provide three level coordination services to the VSC. First, agents can launch requests and publish service to the VSC community through the blackboard of the CA. Second, the CA can collect decisions of agents and check the consistency of interrelated plans. For example, if one manufacturer agent needs two SAs to provide two types of parts, the delivery schedules of the two SAs will be checked by the CA to ensure that it will not delay the manufacturing process. Third, the CA can make group decisions for the VSC to optimize certain performance based on its network knowledge. For example, two SAs send their available-to-promise and inventory information to the CA, which determines delivery plans for SAs.
- ♦ **Integration.** Traditional planning and scheduling functions are distributed in various departments and implemented by heterogeneous application software systems. An MAS adopts wrapper agents to integrate legacy systems, such as planning, scheduling, purchasing, network analysis into a united information

platform.

- ♦ **Reconfiguration.** An agent in an MAS adopts a modular component structure which enables system-level and function-level reconfiguration. From the function-level viewpoint, agents deploy a “Plug-and-Play” technology to build their components. An upgraded problem-solver can be easily installed into an agent to replace the original one without affecting its operation. For example, the mixed linear programming approach can be used for the LA to design optimal transport route around a large region. With the rapid development of computation technology, genetic algorithms have emerged as a promising optimization tool. Thus, the optimization component can be taken out from the application layer of LA and replace it with the new generic algorithm. No further change is required for any other agent or component in this system. From the system-level viewpoint, a VSC is a loose-coupled alliance, where members can join or leave freely without affecting operations of the entire system. VSC can be easily reconfigured to accommodate the introduction of, e.g., new product, facilities, suppliers, new make-or-buy arrangements, and new transportation arrangements, etc.

To achieve the above functionalities, effective negotiation mechanism should be available and a feasible agent platform for real-time negotiation and communication should be chosen.

3.3 Negotiation Mechanism

In a business environment, the functionality of a supply chain management system

Conceptual MAS for VSCM

(SCMS) has been defined as “the right products in the right quantities (at the right place) at the right moment at minimal cost” (Nevem, 1989). In an agent-based system, the functionality of a VSC in an MAS is fulfilled through the negotiation process among functional agents, in which the constraints have been fully or partially satisfied to obtain a common decision.

An MAS consists of heterogeneous types of functional agents. All of such agents have some understanding of the system ontology and use ACL to make conversation. The system ontology includes interaction rules and knowledge about the goods that the system is dealing with.

Broadly, functional agents can be divided into below types:

1) Problem solving agent

Problem solving agent (PSA) is oriented for a unique class of VSCM problems and applied for solving problems in different processes, for example, purchasing, inventory management, etc.

2) System information agent

A system information agent (SIA) is connected with the relevant databases. It is responsible for consistency check, interaction with the outside, and constraints control.

3) Facilitator agent

A facilitator agent (FA) distributes knowledge and identifies access level in sharing data and knowledge base. It has three major components including:

(i) Conventional reusable knowledge management techniques (Stone and Veloso,

1997);

(ii) A common knowledge map that describes distribution of knowledge sources;

and

(iii) Ontology management.

4) Unit agent

A unit agent (UA) represents a certain partner in VSC and implements one special role of the virtual supply chain.

As shown in Figure 3.3, in order to solve a problem, the PSA deploys reusable knowledge management through the FA to communicate with shared object-oriented data models, where are resided in ontology-based/user-oriented knowledge domain models of SIA. The knowledge domain model of a UA describes the environment at the unit level, such as objectives, constraints and resources. An FA utilizes the shared object-oriented data models for checking consistency regarding unit's data and knowledge for problem solving. It also communicates with Internet-based distributed relational databases through object relational adapters.

Figure 3.3 Negotiation mechanism

Conceptual MAS for VSCM

Initially, functional agents might not be aware of who are its cooperating parties in the VSC. However, they can get the information about potential partners from the relevant SIA. The knowledge of each agent is represented as a set of constraints. The negotiation process is modeled as a process of determining values for a set of variables in a collaborative way, for example, agents may negotiate the due dates and quantities of the items. Both the specific dates and quantities are subject to the constraints of suppliers' capacities.

In an MAS, UAs may join the system and leave it according to their own rules or orders from their owners. A UA is said to join the system if it advertises its abilities and desired roles to the SIAs. Similarly, where a UA wants to leave a system, it has to notify SIAs in which it registered. There are no centralized super-agents or distributed mediators (Kalakota et al., 1995) to handle the agent cooperation. All these activities occur through negotiation processes. A system is called "dead" there are either no registered agents or no virtual chains can be formed with a high probability when customer orders arrive.

3.4 Java Agent Development Framework

Many agent systems and toolkits have been developed and described in the literature or on the web. Recently, even numerous Java-based, FIPA compliant systems are seeing use in the wider community. The famous Java-based toolkits are: ZEUS and JADE (Nwana, 1999). In this thesis, JADE is chosen as the platform to implement the proposed agent-based system for VSCM.

JADE (Java Agent Development Framework) is a software development framework

Conceptual MAS for VSCM

aimed at developing multi-agent systems and applications conforming to FIPA standards for intelligent agents. It includes two main products: a FIPA-compliant agent platform and a package to develop Java agents. JADE has been fully coded in Java and an agent programmer. Table 3.1 summarizes the JADE main characteristics.

Table 3.1 JADE technical details

Name	JADE - Java Agent Development Framework
Provider	TILAB
Web site	http://jade.tilab.com/
Contact point	Fabio Bellifemine, email: fabio.bellifemine@tilab.com
Language	Java: J2EE, J2SE, J2ME CLDC/MIDP1.0 platforms
Availability	Open Source, LGPL license If needed, commercial licenses for specific purposes or consultancy frameworks can be properly negotiated.
Technical/functional characteristics	Distributed, multi-party application with peer-to-peer communication. Compliance with the FIPA standard. Agent life cycle management. White pages and yellow pages services with the opportunity of creating federation graphs at run-time. Graphical tools supporting the debugging, management and monitoring phases. Support for agent code and execution state migration. Support for complex interaction protocols (e.g. contract-net). Support for messages content creation and management including XML and RDF. Support for integration in JSP pages by means of a tag library. Support for application level security (currently only in J2SE). Transport protocols selectable at run-time. Currently available: JAVA-RMI, JICP (JADE proprietary protocol), HTTP and IIOP.
Network environment	Already tested in the fields over Bluetooth, GPRS, W-LAN and the Internet.
Terminals	All terminals supporting Java MIDP1.0 or Personal Java or J2SE. Already tested on Nokia 3650, Motorola Accompli008, Siemens SX45, PalmVx, Compaq iPaq, Psion5MX, HP Jornada 560.

JADE is written in Java language and is made of various Java packages, giving application programmers both ready-made pieces of functionality and abstract interfaces for custom, application dependent tasks. Java was the programming language of choice because of its many attractive features, particularly geared towards object-oriented programming in distributed heterogeneous environments; some of these features are Object Serialization, Reflection API and Remote Method Invocation (RMI).

3.4.1 JADE platform

The standard model of an agent platform, as defined by FIPA, is represented as Figure 3.4.

Figure 3.4 Architecture of a FIPA agent platform

The Agent management system (AMS) is the agent who exerts supervisory control over access to and use of the Agent Platform. Only one AMS will exist in a single platform. The AMS provides white-page and life-cycle service, maintaining a directory of agent identifiers (AID) and agent state. Each agent must register with an AMS in order to get a valid AID. The directory facilitator (DF) is the agent who provides the default yellow page service in the platform. The message transport system, also called agent communication channel (ACC), is the software component controlling all the exchange of messages within the platform, including messages to/from remote platforms.

JADE fully complies with the FIPA agent platform, and when a JADE platform is launched, the AMS and DF are immediately created and the ACC module is set to allow message communication. The agent platform can be split on several hosts (see

Conceptual MAS for VSCM

Fig 3.5). Only one Java application, and therefore only one Java virtual machine (JVM), is executed on each host. Each JVM is a basic container of agents that provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host. The main-container, or front-end, is the agent container where the AMS and DF lives and where the RMI registry, that is used internally by JADE, is created. The other agent containers, instead, connect to the main container and provide a complete run-time environment for the execution of any set of JADE agents.

Figure 3.5 JADE agent platform distributed over several containers

3.4.2 Agent communication

The class `ACLMessage` represents ACL messages that can be exchanged between agents. It contains a set of attributes as defined by the FIPA specifications.

The JADE platform presents a single interface to the outside world using standard ACC agent. This agent is a CORBA IIOP server object and listens for remote invocations. Every time it receives an ACL message encoded as a string (usually by

Conceptual MAS for VSCM

non-JADE agents), it parses the message and converts it into a Java ACLMessage object used by all JADE agents. It can also perform the dual conversion when a JADE agent sends a message to a non-JADE agent. The agent container allows JADE to conform the communication mechanisms that must be unknown to all the agents. In fact, every agent thread is a completely independent class, which does not know the details of communication mechanisms and agent platform architecture

An agent willing to send a message should create a new ACLMessage object, fill its attributes with appropriate values, and finally call the method *Agent.send()*. Likewise, an agent willing to receive a message should call *receive()* or *blockingReceive()* methods, both implemented by the Agent class. Sending or receiving messages can also be scheduled as independent agent activities by adding the behaviours *ReceiverBehaviour* and *SenderBehaviour* to the agent queue of tasks.

All the attributes of the ACLMessage object can be accessed via the *set/get<Attribute>()* access methods. All attributes are named after the names of the parameters, as defined by the FIPA specifications. Those parameters whose type is a set of values (like receiver, for instance) can be accessed via the methods *add/getAll<Attribute>()* where the first method adds a value to the set, while the second method returns an *Iterator* over all the values in the set. Notice that all the *get()* methods return null when the attribute has not been yet set.

Furthermore, this class also defines a set of constants that should be used to refer to the FIPA performatives, i.e. *request*, *inform* etc. When creating a new ACLMessage

Conceptual MAS for VSCM

object, one of these constants must be passed to `ACLMessage` class constructor, in order to select the message performative. The `reset()` method resets the values of all message fields. The `toString()` method returns a string representing the message. This method should be just used for debugging purposes.

When a JADE agent sends a message, the following different cases are possible:

- If the receiver agent lives on the same agent container, the Java object representing the ACL message is passed to the receiver using an event object, without any message translation, for example, the message sent by Agent 1 to Agent 2 in figure 3.6.
- If the receiver agent lives on the same JADE platform but within a different container, the ACL message is sent using Java Remote Method Invocation framework for distributed object computing. Java RMI allows transparent object marshalling and unmarshalling, avoiding tedious message conversion. Apart from performance, the agent receives a Java object, just like intra-container messaging, for example, the message sent by Agent 3 to Agent 2 in figure 3.6. In this case, a cache hit is supposed to occur, so platform front-end is not contacted.
- If the receiver lives on a different agent platform, standard IIOP protocol and OMG IDL interface are used, according to FIPA standard. This involves translating ACL message object into a character string and then performing a remote invocation using IIOP as middleware protocol. On receiver side, an IIOP unmarshalling will occur, yielding a Java String object, which will be parsed into an `ACLMessage` object. Eventually, the Java object will be dispatched to the receiver agent (via Java events or RMI calls).

Conceptual MAS for VSCM

Figure 3.6 Communication between agents

JADE runtime maintains suitable agent tables and is thus able to choose the most efficient messaging mechanism, according to receiver agent location. Besides, address caching is used on each container to avoid looking up platform front-end Global Agent Descriptor Table all the way.

Chapter 4 Proposed Negotiation-based MAS

With consideration of the lifecycle of a VSC and the characteristics of agent-based supply chain systems, this chapter proposes a negotiation-based MAS (NBMAS) for VSCM. The focused negotiation process is presented first. Following are the many issues regarding the NBMAS, including a system architecture, databases and negotiation rules.

4.1 Negotiation Process

Figure 4.1 shows the negotiation process between a purchaser in a company and its suppliers.

Figure 4.1 Negotiation process

When the company receives the order from customer, it analyzes and decomposes the order into list containing the relevant components. All of the material components of the product placed in the order is listed. In addition, other information regarding

Proposed Negotiation-based MAS

quantity, delivery lead time and cost of these components are also included in the list. Upon receiving the list, the purchaser sends enquiry to suppliers to find the feasible ones that can meet the requirements of these components. Based on their inventories, production capabilities and manufacturing resource availabilities, each supplier responds to the purchaser differently. If a supplier can not meet the delivery requirements of certain components, he rejects the purchaser's enquiry. If he can fulfill the order, he sends the corresponding quotation to purchaser. After receiving the quotations, the purchaser starts to compare costs and delivery lead time of each quotation. Based on the compared result, he will select the optimal supplier from these feasible ones. At last, the purchaser sends the purchase order to the selected supplier.

4.2 System Architecture

Figure 4.2 shows the architecture of the proposed negotiation-based MAS (NBMAS) for VSCM.

Proposed Negotiation-based MAS **Figure 4.2 System architecture**

A customer agent generates a reference number for each order. It also stores order configuration in terms of orders for constituting materials into the database and sends a *query-ref* ACL message to the configuration agent. After receiving orders from customer agent, the configuration agent decomposes orders into orders for sub-components. Acquire such information about current manufacturing capacities and warehouse level, and eventually generate the bill of material (BOM) to be sent to a purchasing agent. Upon receiving an order from the configuration agent, the purchasing agent starts communicating with a number of suppliers for determining an optimal one.

First, the purchasing agent will check the agent databases to identify potential suppliers. Subsequently, it will negotiate with the suppliers directly and select the

Proposed Negotiation-based MAS

most suitable one. When receiving the purchase request from purchasing agent, the supplier agent checks its inventory database. If the inventory is enough to offer the required items, it will send the response to the purchasing agent.

4.2.1 Databases

Several databases are introduced in the proposed model for providing the necessary data and information to the agents. They include a logistics database, a purchasing database and a supplier database.

Logistics database

This database provides warehouse and transporter profiles and data for agents to determine the suitable suppliers for different orders. Such data are included in this database: products, service, locations, prices and availabilities.

Purchasing database

This database contains sub-system catalogues, products catalogues and components catalogues. Some information, such as special delivery requirements of certain products is also included in the database. The data in this database can be classified into five types as follows:

- 1) Product name: the items which customers want to buy;
- 2) Sub-component name: the end-items comprising the products;
- 3) Amount: the amounts of the sub-components;
- 4) Expected price: acceptable prices of the sub-components; and
- 5) Due date: the delivery date.

Proposed Negotiation-based MAS

Supplier database

This database contains material lists of different vendors and vendor's delivery capacity. These data will be used for suppliers to determine whether they can meet the order requirements. The data in this database includes:

- 1) Component name: the catalogues of components that a supplier can offer;
- 2) Component price: the prices of components;
- 3) Delivery date: delivery lead times;
- 4) Component amount: the amounts of the components that a supplier holds; and
- 5) Transportation cost: the costs of transporting items from a supplier to a buyer.

Besides these databases, a log is also necessary for an agent to improve its performance and to generate process reports. The information and data in the log is updated constantly by warehouses, transporters and agents within the environment.

4.2.2 Negotiation and selection rules

To determine optimal suppliers, two types of rules, including negotiation rules and selection rules, are incorporated in the negotiation mechanism.

Negotiation rule

These rules are used to identify the available suppliers. The purchasing agent sends the purchase requests to the supplier agent. A purchase request may contain the data in purchasing database, e.g. sub-component name, expected price. Then, the suppliers will go to their databases to check if they have enough capacities to offer

Proposed Negotiation-based MAS

the items to the purchaser. Available suppliers indicate that they can satisfy all the requirements from purchaser, e.g., component price *less than* expected price.

Selection rule

These rules are used to determine the optimal supplier from the available suppliers.

In doing so, these rules are constructed incorporate five parameters as follows:

- 1) Cost: the objective is to minimize the cost;
- 2) Time: the objective is to minimize the lead times;
- 3) Location & Delivery: the objective is to choose the shortest route;
- 4) Quality of Service: the objective is to choose products with the highest quality; and
- 5) Credibility of Parties: this considers the credibility involved.

Chapter 5 Case Study

5.1 Company Background

The proposed NBMSA for VSCM has been tested in an electronics company XYZ (disguised name). XYZ is a multinational corporation and has a number of design centers, manufacturing plants, warehouses and suppliers located in different countries. In 2003, XYZ fulfilled customer orders for 37870 products/items. Due to the dispersed locations of its suppliers, manufacturing plants, distribution centers (DCs) and warehouses, many issues are raised. To obtain the most added values, XYZ is struggling to select proper suppliers, manufacturing plants, warehouses, and DCs so as to form different supply chains in response to different customer orders. Previously, XYZ made the selection decisions based on the experience and subjective knowledge of the supply chain managers without a rigorous and theoretical sounded system or mechanism.

Motor is the main product family of XYZ Company. Here, motors are selected as the tested parts to build the prototype. Figure 5.1 shows the main assemblies and components of motors.

Figure 5.1 Motor description

A motor consists of two major assemblies, including a driver assembly and a case assembly. Each assembly is formed by two components. A driver assembly is formed by rotor and stator. A case assembly is formed by shell and base.

5.2 Business Process

While the business process of XYZ is rather complex, this case application adopts a simplified purchase process for illustration. Figure 5.2 shows the four major steps involved in the process.

Figure 5.2 Purchase process

1. Customer order decomposition

For each customer order, XYZ decomposes the placed motor into the corresponding assemblies and components. Based on the ordered quantity of a motor and the inventory information, the quantities for the decomposed assemblies and components are determined. Also determined are the lead times and prices requirements for the assemblies and components. Consequently, purchase requisition is generated based on above information and sent to the suppliers.

2. Purchase requisition sending to suppliers

Based on above step, XYZ sends the purchase requisition to its 7 suppliers. The purchase requirements are made according to the prices, lead times and quantities. Table5.1 shows the requirements of items for motor A placed in purchase requisition. For example: a sub-component stator2 with price lower than 80.

Table 5.1 Purchase requisition

Product: motorA			
Sub-component name	Expected price	Expected lean time	Expected amount
Sub-component1: rotor1	50	8	1000
Sub-component2: stator2	80	8	1000
Sub-component3: shell1	75	10	3000
Sub-component4: base1	100	10	3000

3. Supplier selection

The whole selection process includes following two steps:

Case Study

1) **Feasible supplier selection:** Upon receiving the purchase requests, XYZ’s suppliers check their own items information to identify if they have the enough capacity to meet the purchase requisition. Then the suppliers send their response (available or unavailable) to XYZ. Based on the responses, the feasible suppliers are identified. The specific information pertaining to the items of the 7 suppliers is given in Table 5.2—Table 5.7.

Table 5.2 Supplier1 information

supplier1				
Sub-component name	unit price	lean time	amount	Transit-cost
rotor1	80	7	7000	500
rotor2	75	7	7000	500
stator1	90	9	7000	500
shell1	70	9	7000	500
shell2	70	8	7000	500
shell3	70	8	7000	500
base2	90	8	7000	500

Table 5.3 Supplier2 information

supplier2				
Sub-component name	unit price	lean time	amount	Transit cost
rotor1	45	7	7000	500
rotor2	40	7	7000	500
stator2	80	7	7000	500
shell1	70	7	7000	500
shell2	70	7	7000	500
shell3	70	7	7000	500
base1	90	7	7000	500

Table 5.4 Supplier3 information

supplier3				
Sub-component name	unit price	lean time	amount	Transit-cost
rotor1	60	10	7000	400
rotor2	60	10	7000	400
stator1	90	10	7000	400
shell1	70	10	2000	400
shell2	70	10	2000	400
shell3	70	10	2000	400
base2	90	10	7000	400

Case Study

Table 5.5 Supplier4 information

supplier4				
Sub-component name	unit price	lean time	amount	Transit-cost
rotor1	40	5	7000	650
rotor2	60	5	7000	650
stator1	90	5	7000	650
stator2	70	5	2000	650
shell2	70	12	2000	650
shell3	70	12	2000	650
base2	90	12	7000	650

Table 5.6 Supplier5 information

supplier5				
Sub-component name	unit price	lean time	amount	Transit-cost
rotor1	40	6	5000	500
stator1	60	6	5000	500
stator2	90	6	5000	500
stator3	90	6	5000	500
shell2	60	6	5000	500
shell3	70	6	5000	500
base2	90	8	5000	500

Table 5.7 Supplier6 information

supplier6				
Sub-component name	unit price	lean time	amount	Transit-cost
rotor1	40	6	6000	550
stator1	40	6	6000	550
stator2	80	6	6000	550
shell1	70	6	6000	550
shell2	70	6	6000	550
base1	70	6	6000	550
base2	90	6	6000	550

Table 5.8 Supplier7 information

supplier7				
Sub-component name	unit price	lean time	amount	Transit-cost
rotor1	40	9	7000	500
rotor2	60	9	7000	500
stator1	60	9	7000	500
stator2	60	9	7000	500
shell2	60	12	5000	500
shell3	70	12	5000	500
base2	90	12	5000	500

2) **Optimal supplier selection:** If several suppliers can fulfill the purchase requisition, selection methods should be developed by XYZ to determine the optimal one. Then, purchase order is sent to the optimal supplier. So many methods can be

designed as selection rules. Here, a mixed linear objective function is created as follows:

$$V = \arg \min \sum_{i=1}^n w_i \times c_i, \quad \text{wrt} \quad \sum_{i=1}^n w_i = 1$$

In this function, w_i is the weight and c_i is the condition. Normally w_i s more affects the target V than c_i s. Condition c_i can represent any quantitative factor such as the price or the lean time. The choice of c_i relies on the company policy. Target factor V is the weighted combination of the c_i s. The optimal supplier is the one having the minimum V .

4. Order purchase

Purchaser sends the purchase order to the optimal supplier. Then the business flow entries into the next stage, order processing.

5.3 Implementation

The prototype is coded in Java. The open source JADE was adopted as the communication platform.

In the prototype, two agents including *BuyerAgent* and *SupplierAgent* are developed. They represent XYZ and its suppliers respectively. All the data in the purchase requisition and supplier information are represented as XML-format text databases.

5.4 Execution

1. Purchaser receives the customer order from the internet while the system decomposes the end production into brief information of sub-components automatically. Figure 5.3 shows the brief information of Motor A. Before creating the purchase requisition, purchaser has to refer to the historical record of sub-components in details.

Figure 5.3 Customer order information

2. Purchasing requisition is generated based on the detailed information of items, e.g., current components quantities in inventory, average prices. Figure 5.4 shows the detailed information of a sub-component.

Field	Value
Material number	T-M516
Material description	Stator2
Quantity in inventory	600 PC
Storage location	1000
Movement type	101
Stock type	Unrestricted-use
Moving average price	\$0

Figure 5.4 Detailed information of component

Case Study

The name of the purchase requisition order is “Motor A”. Then the BuyerAgent will send this enquiry to its seven suppliers.

3. After completing setting up of running environment, the prototype first launched the GUI of main container as shown in Figure 5.5.

Prompt> java jade. Boot – gui

Figure 5.5 The starting of JADE platform

4. To XYZ, then adds the agent, the results are shown in Figure 5.6. Seven suppliers are added in the GUI, they are created by the SupplierAgent. One purchaser of XYZ is added in the GUI, it is created by the BuyerAgent

Figure 5.6 JADE GUI for agents

Case Study

5. The BuyerAgent sends an ACL message to the seven suppliers. This ACL message includes negotiation rules that are defined enough to the order information. Only the *query-ref* communication act of the message is acceptable by the SupplierAgent. The content of the message must be “Motor A”. According to this content, the system links to the relevant database automatically. This process is shown in Figure 5.7 and Figure 5.8.

Figure 5.7 ACL message GUI

Figure 5.8 The result

Suppliers who can satisfy the negotiation rules will response to the BuyerAgent. The linear selection criterion will be applied to feasible suppliers. At last, an optimal supplier is determined as shown in Figure 5.9. The corresponding supplier’s database

will be updated accordingly.

Figure 5.9 System outcome of the process

5.5 Evaluation

To verify if the prototype generates the accurate optimal supplier, evaluation is performed.

- 1) Feasible supplier selection. All the information of seven suppliers is contrasted with the information of purchase requisition. Figure 5.10 shows how supplier2 and supplier6 are chosen as the feasible suppliers.

Figure 5.10 Feasible suppliers selection

2) Optimal supplier selection. A linear selection criterion should be defined as the cost function to evaluate responded suppliers in the purchaser end. The linear selection criterion is given by:

$$\begin{aligned} \text{Cost} = & \left(\sum_{i=1}^4 0.7 \times \text{subcomponent}_i _ \text{unitprice} \times \text{subcomponent}_i _ \text{amount} \right) + \\ & \left(\sum_{i=1}^4 0.15 \times \text{subcomponent}_i _ \text{leantime} \right) + \\ & \left(\sum_{i=1}^4 0.15 \times \text{subcomponent}_i _ \text{transit_cost} \right) \end{aligned}$$

Afterwards, calculate the cost value for each feasible supplier. The detained calculation process is given by:

$$\begin{aligned} \text{Cost (Supplier2)} &= 0.7 \times (1000 \times 45 + 1000 \times 80 + 3000 \times 70 + 3000 \times 90) + 0.15 \times 7 + 0.15 \times 500 \\ &= 423576.1 \end{aligned}$$

$$\begin{aligned} \text{Cost (Supplier6)} &= 0.7 \times (1000 \times 40 + 1000 \times 80 + 3000 \times 70 + 3000 \times 70) + 0.15 \times 6 + 0.15 \times 550 \\ &= 378083.4 \end{aligned}$$

Thus, supplier6 with a smaller cost value of 378083.4 is the optimal supplier, which justifies the correctness of the result, as shown in Figure 5.9.

Chapter 6 Conclusions and Recommendations

6.1 Conclusions

In real business environment, it is so difficult to select the most qualified ones among multiple suppliers and form the dynamic VSC according to the different requirements of different customer orders. These issues raise the importance in developing a negotiation mechanism for companies to negotiate and communicate with possible partners effectively. In this thesis, the proposed model of NBMAS for VSCM is designed and implemented via using JADE. The results show the effectiveness and efficiency of the proposed system. It also hints at its practicality, and thus paves the way for additional future work.

6.2 Contributions

The NBMAS has both academic significance and practical relevance.

Academic Significance

- 1) The proposed model for VSCM overcomes the limitations of existing systems for SCM.
- 2) The application of JADE stimulates more applications in building systems..
- 3) The proposed system provides a useful paradigm for further complex system design.

Conclusions and Recommendations

Practical Relevance

- 1) Shed light on efficient negotiation process. Companies can develop the one that suit them.
- 2) Assist companies in selecting proper partners to response demand changes.
- 3) Contribute to develop long-term customer-supplier relationships.

6.3 Limitations and Future Work

The main limitation of the proposed multi-agent VSC system lies in its reliance on the message communication mechanisms of JADE, a freeware in Internet. Therefore, lack of security concerns for the real business flaw in logistic management, which may not be acceptable in the real world. This means that in practice, the logistic information is at a risk of being monitored by unexpected persons. The relaxation of this reliance involves significant challenges.

In chapter 5, the case only implements small function of the VSC network. A lot more complex processes are more likely to be used in the real world. The future research subject will be how to formalize a variety of dynamics and methodology to deal with more complex situation and to extend the scope to a broader supply chain environment. So many other methods can be used to decide the optimal supplier, just like genetic algorithm (GA). More selection rules can be explored in more systematic and scientific way.

Security and threats are very critical points for such system, where each agent can be

Conclusions and Recommendations

a virtual enterprise and has its own rules for security that may restrict other parts of the chain from accessing their resources in different way. Current state-of-the-art technologies will not be able to guarantee the absolute security in the foreseeable future. But incremental improvements can be made in multi-agent VSC system by incorporating more sophisticated distributed computing techniques such as the CORBA distributed architecture.

Reference

Aken, J., The virtual organizational: a special mode of strong inter-organizational cooperation. *Managing Strategically in an Interconnected World*, pp. 301-320, 1998

Ambroszkiewicz, S., Cetnarowicz, K. and Radko, B., Enterprise formation mechanisms based on mobile agents. *Intelligent Agents in Information and Process Management*, No.9, pp. 23-34, 1998

Bellifemine et. al. *JADE Administrator's Guide*. Torino. TILAB, 2002

Bellifemine et. al. *JADE Programmer's Guide*. Torino. TILAB, 2002

Bongaerts, L., Valckenaers, P., Van, H. and Wyns, J., Schedule execution for a holonic shop floor control system. *The ASI-95 of NOE on ICIMS*, Lisboa, pp.115-124, June, 1995

Bussmann, S., An agent-oriented architecture for holonic manufacturing controls. *The 1st Int. Workshop on Intelligent Manufacturing Systems*, EPFL, Lausanne, Switzerland, pp. 1-12, 1998

Byrne, J., The virtual corporation, *Business Week*, 8, Feb, 1993

Chandra, C., Enterprise architectural framework for supply chain integration, *The 6th Annual Industrial Engineering Research Conference*, Miami Beach, Florida, Institute of Industrial Engineers, Norcross, Georgia, pp. 873-878, May, 1997

Clements, P. and Papaioannou, T., Enabling the virtual enterprise. *Engineering Logistics and Achievement*. Mechanical Engineering Publications Ltd., July, 1997

Cloutier, L., Frayret, J., Amours, B., Espinasse and Montreuil, B., A commitentoriented approach to networked manufacturing. *IEPM'99 International*

Reference

Conference on Industrial Engineering and Production Management, pp. 487-496, July, 1999

Davidow, M. and Malone, M., *The Virtual Corporation: Structuring and Revitalizing the Corporation for the 21st Century*. Harper Business, 2002

Durfee, E., Lesser, V. and Corkill, D., Cooperative distributed problem solving. In Barr, *The Handbook of Artificial Intelligence*, Vol. 4, pp. 83-148, 1989

FIPA Web Site, <http://www.fipa.org>

Fischer, K., Muller, J., Heimig, I. and Scheer, A., Intelligent agents in virtual enterprises. *The Practical Applications of Intelligent Agents and Multi-Agent Technology*, London, U.K. pp. 205-224, April, 1996

Fox, M. and Gruninger, M., Enterprise Modeling, *AI Magazine*, AAAI Press, pp. 109-121, 1998

Fox, M.S., Barbuceanu, M., and Teigen, R., Agent oriented supply chain management, *The International Journal of Flexible Manufacturing System*, vol. 12, pp. 165-188, 2000

Fox, S., Chionglo, F. and Barbuceanu, M., *The Integrated Supply Chain Management*, University of Toronto, December, 1993

Genesereth, M. R. and Ketchpel, S, Software Agents, *Communications of the Association for Computing Machinery*, July, 1994

Gilman, R., Aparicio, M., Barry, J., Durniak, T., Lam, H. and Ramnath, R., Integration of design and manufacturing in a virtual enterprise using enterprise rules. *Intelligent Systems for Manufacturing Integration*, pp. 160--171, 1997

Reference

Handfield, B. and Nichols, L., Introduction to supply chain management, *Prentice Hall*, 1999

Jagdev, H. and Browne, J., The extended enterprise: a context for manufacturing. *Production Planning & Control*, Vol.9.No.3, pp. 216-229, 1998

Kalakota, R., Stallaert, J. and Whinston, A. B., Implementing real time supply chain optimiaztion system, *The Conference on Supply Chain Management*, Hong Kong, 1995

Kjenstad, D., Coordination supply chain scheduling, Ph. D. Dissertation, Department of Production and Quality Engineering, Norwegian University of Science and Technology, 1998

Labrou, Y., Finin, T. and Y. Peng, The current landscape of Agent Communication Languages, *Intelligent Systems*, Vol.14.No. 2, 1999

Rosenschein, J., Rational interaction: cooperation among intelligent agents. PhD thesis, Stanford Universit, 2000

Sadeh, N., An agent architecture for multi-Level mixed initiative Supply Chain Coordination, Internal Report, *Intelligent Coordination and Logistics Laboratory*, Carnegie Mellon University, 1996

Sahingoz, O. and Erdogan, N., A two leveled mobile agent system for electronic commerce, *Aeronautics and Space Technologies*, pp. 21-32, 2003

Shaw, M. and Whinston, A., Distributed planning in cellular flexible manufacturing systems. *Management Information Research Center*, Purdue University, 1983

Reference

Sikora, R. and Shaw, M., A multi-agent framework for the coordination and integration of information systems. *Management Science* Vol.44.No.11, pp. 65-78, 1998

Simatupang, M. C., Wright and Sridharan, R., The knowledge of coordination for supply chain integration, *Business Process Management Journal*, Vol.8.No.3, 2002

Smirnov, C., Virtual supply chain management: information framework and agreement network model, Computer Supported Cooperative Work in Design, *The 6th International Conference*, 2001

Smith, R., The contract net protocol; High-level communication and control in a distributed problem solver, *IEEE Transaction on Components*, IEEE Press. pp. 1104-1113, 1980

Sousa, P. T., Heikkila, M. Kollingbaum, and Valckenaers, P., Aspects of co-operation in Distributed Manufacturing Systems, *The 2nd International Workshop on Intelligent Manufacturing Systems*, Leuven, Belgium, pp. 685-717, September, 1999

Stone, P. and Veloso, M., Multi agent Systems: A Survey from a Machine Learning Perspective”, *Journal publication Review*, February, 1997

Sycara, K., Multi agent compromise via negotiation, *Distributed Artificial Intelligence*, L. Gasser and M.N.Huhns, Eds, Vol. 2, pp. 119-137, 1989

Symeonidis, A. L., Kehagias, D., Koumpis, A. and A. Vontas, Open Source Supply Chains, *10th International Conference on Concurrent Engineering (CE-2003)*, Madeira, Portugal, 2003

Tarokh, H. and Tayebi, A., Mobile agent based intelligent supply chain management”, *IEEE SOLI Conference*, 2005

Reference

Umeda, S. and Jones, A., Virtual supply chain management: a re-engineering approach using discrete event simulation, *World Conf. SCI '97 CARACAS*, 1997

Wagner, T., Guralnik, V. and Phelps, J., Software agents: Enabling Dynamic Supply Chain Management for a Build to Order Product Line, *Honeywell Laboratories*, 2002

Wooldridge, M. and Jennings, N., Intelligent agents: theory and practice, *Knowledge Engineering Review* Vol.10.No.2, pp. 115, 1995

Appendix

1. Source Code of BuyerAgent:

```

/*****
import jade.core.Agent;
import jade.core.AID;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.domain.DFService;
import jade.domain.FIPAAException;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;

import java.io.*;
import java.util.Vector;
import yukki_master_project.XMLReader;

public class OrderAgent extends Agent {
    // The list of known seller agents
    private AID[] supplierAgents;
    //define an xmlReader
    private XMLReader xmlReader = new XMLReader();
    //define the message channel
    ACLMessage msg;

    private int NUM_Components = 4;
    private int NUM_Attr = 3;

    class WaitRequestAndReplyBehaviour extends SimpleBehaviour {
        private boolean finished = false;
        public WaitRequestAndReplyBehaviour(Agent a) {
            super(a);
        }
        public void action() {
            msg = blockingReceive();
            if(msg != null) {
                try {
                    System.out.println("Received the following order: "+ msg.getContent());
                }
                catch(Exception e) {
                }
            }

            ACLMessage reply = msg.createReply();
            if(msg.getPerformative() == ACLMessage.QUERY_REF) {
                String content = msg.getContent();
                if (content != null) {
                    process(msg.getContent());
                    reply.setPerformative(ACLMessage.INFORM);
                    reply.setContent("Affirmative");
                }
            }
            else {
                reply.setPerformative(ACLMessage.NOT_UNDERSTOOD);
                reply.setContent("( UnexpectedContent (expected != 0))");
            }
        }
    }
}

```

```

    }
    else {
        reply.setPerformative(ACLMessage.NOT_UNDERSTOOD);
        reply.setContent("(" + ACLMessage.getPerformative(msg.getPerformative()) + ") (expected (query-ref :content != 0)))");
    }

    try {
        System.out.println("Sending the following message: " + reply.getContent());
    }
    catch (Exception e) {
    }

    send(reply);
}
else {
    System.out.println("No message received");
}
}
public boolean done() {
    return finished;
}
} //End class WaitRequestAndReplyBehaviour

//read database and call supplier agents
protected void process(String dbname) {
    //read the order xml database
    String orderBuf = "";
    try {
        xmlReader.readFile(dbname + ".txt");
    } catch (Exception e) {
    }

    orderBuf = xmlReader.extractField("<order>", "</order>", orderBuf);
    // Get the order details
    final String motorType = xmlReader.extractField("<type>", "</type>", orderBuf);
    //Vector order structure:
    //motorType
    //component
    //    name
    //    attribute
    //        attributes for every component: expected price, expected lean time, expected
amount
    final Vector order = new Vector();
    order.add(motorType);
    for (int i = 0; i < NUM_Components; i++) {
        String startToken = "<subcomponent id=\"" + i + "\">";
        String stopToken = "<subcomponent>";
        String componentBuf = xmlReader.extractField(startToken, stopToken, orderBuf);
        Vector component = new Vector();
        String name = xmlReader.extractField("<name>", "</name>", componentBuf);
        component.add(name);
        for (int j = 0; j < NUM_Attr; j++) {
            startToken = "<attribute id=\"" + j + "\">";
            stopToken = "<attribute>";
            String attrBuf = xmlReader.extractField(startToken, stopToken, componentBuf);
            Vector attr = new Vector();
            String attrName = xmlReader.extractField("<name>", "</name>", attrBuf);

```

Appendix

```

        String attrRelation = xmlReader.extractField("<relation>", "</relation>", attrBuf);
        String attrAmount = xmlReader.extractField("<amount>", "</amount>", attrBuf);
        attr.add(attrName);
        attr.add(attrRelation);
        attr.add(attrAmount);
        component.add(attr);
    }
    order.add(component);
}

if ((motorType.trim().compareTo("") != 0) && order.size() > 1) {
    System.out.println("Target motor type is " + motorType);
    // Add a TickerBehaviour that schedules a request to seller agents every minute
    addBehaviour(new TickerBehaviour(this, 60000) {
        protected void onTick() {
            System.out.println("Trying to order " + motorType);
            // Update the list of supplier agents
            DFAgentDescription template = new DFAgentDescription();
            ServiceDescription sd = new ServiceDescription();
            sd.setType("motor-supplier");
            template.addServices(sd);
            try {
                DFAgentDescription[] result = DFService.search(myAgent, template);
                System.out.println("Found the following supplier agents:");
                supplierAgents = new AID[result.length];
                for (int i = 0; i < result.length; ++i) {
                    supplierAgents[i] = result[i].getName();
                    System.out.println(supplierAgents[i].getName());
                }
            } catch (FIPAException fe) {
                fe.printStackTrace();
            }

            // Perform the request
            myAgent.addBehaviour(new RequestPerformer());
        }
    });
}
else {
    // Make the agent terminate
    System.out.println("No target motor type specified");
    doDelete();
}
}

// Put agent initializations here
protected void setup() {
    // Printout a welcome message
    System.out.println("Hello! Order-agent "+getAID().getName()+" is ready.");
} //ending of setup()

// Put agent clean-up operations here
protected void takeDown() {
    // Printout a dismissal message
    System.out.println("Order-agent "+getAID().getName()+" terminating.");
}

* Inner class RequestPerformer.

```

Appendix

```

* This is the behaviour used by order agents to request suppliers
**/
private class RequestPerformer extends Behaviour {
    private AID bestSupplier; // The agent who provides the best offer
    private int bestPrice; // The best offered price
    private int repliesCnt = 0; // The counter of replies from seller agents
    private MessageTemplate mt; // The template to receive replies
    private int step = 0;

    public void action() {
        switch (step) {
            case 0:
                // Send the cfp to all suppliers
                ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
                for (int i = 0; i < supplierAgents.length; ++i) {
                    cfp.addReceiver(supplierAgents[i]);
                }

                cfp.setContent("haha");
                cfp.setConversationId("book-trade");
                cfp.setReplyWith("cfp"+System.currentTimeMillis()); // Unique value
                myAgent.send(cfp);
                // Prepare the template to get proposals
                mt =
                MessageTemplate.and(MessageTemplate.MatchConversationId("book-trade"),
                                    MessageTemplate.MatchInReplyTo(cfp.getReplyWith()));

                step = 1;
                break;
            case 1:
                // Receive all proposals/refusals from supplier agents
                ACLMessage reply = myAgent.receive(mt);
                if (reply != null) {
                    // Reply received
                    if (reply.getPerformative() == ACLMessage.PROPOSE) {
                        // This is an offer
                        int price = Integer.parseInt(reply.getContent());
                        if (bestSupplier == null || price < bestPrice) {
                            // This is the best offer at present
                            bestPrice = price;
                            bestSupplier = reply.getSender();
                        }
                    }
                    repliesCnt++;
                    if (repliesCnt >= supplierAgents.length) {
                        // We received all replies
                        step = 2;
                    }
                }
                else {
                    block();
                }
                break;
            case 2:
                // Send the purchase order to the supplier that provided the best offer
                ACLMessage orderMsg =
                new
                ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
                orderMsg.addReceiver(bestSupplier);
                orderMsg.setContent("haha");

```



```

        orderMsg.setConversationId("book-trade");
        orderMsg.setReplyWith("order"+System.currentTimeMillis());
        myAgent.send(orderMsg);
        // Prepare the template to get the purchase order reply
        mt
MessageTemplate.and(MessageTemplate.MatchConversationId("book-trade"),
                    MessageTemplate.MatchInReplyTo(orderMsg.getReplyWith()));
        step = 3;
        break;
    case 3:
        // Receive the purchase order reply
        reply = myAgent.receive(mt);
        if (reply != null) {
            // Purchase order reply received
            if (reply.getPerformative() == ACLMessage.INFORM) {
                // Purchase successful. We can terminate
                System.out.println("haha"+" successfully purchased from agent " +
reply.getSender().getName());
                System.out.println("Price = " + bestPrice);
                myAgent.delete();
            }
            else {
                System.out.println("Attempt failed: requested book already sold.");
            }
            step = 4;
        }
        else {
            block();
        }
        break;
    }
}

public boolean done() {
    if (step == 2 && bestSupplier == null) {
        System.out.println("Attempt failed: " + "haha" + " not available for sale");
    }
    return ((step == 2 && bestSupplier == null) || step == 4);
}
} // End of inner class RequestPerformer
}

```

2. Source Code of SupplierAgent:

```

package yukki_master_project;

import jade.core.Agent;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.domain.DFService;
import jade.domain.FIPAAException;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;

import java.util.*;
import yukki_master_project.XMLReader;

```

Appendix

```

public class SupplierAgent extends Agent {
    // The catalogue of books for sale (maps the title of a book to its price)
    private Hashtable catalogue;
    // The GUI by means of which the user can add books in the catalogue
    private SupplierAgentGUI myGui;
    //define an xmlReader
    private XMLReader xmlReader = new XMLReader();

    // Put agent initializations here
    protected void setup() {
        // Create the catalogue
        catalogue = new Hashtable();
        // Register the supplier service in the yellow pages
        DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(getAID());
        ServiceDescription sd = new ServiceDescription();
        sd.setType("motor-supplier");
        sd.setName("JADE-motor-trading");
        dfd.addServices(sd);
        try {
            DFService.register(this, dfd);
        }
        catch (FIPAException fe) {
            fe.printStackTrace();
        }

        // Add the behaviour serving queries from order agents
        addBehaviour(new OfferRequestsServer());
        // Add the behaviour serving purchase orders from order agents
        addBehaviour(new PurchaseOrdersServer());
    }

    // Put agent clean-up operations here
    protected void takeDown() {
        // Deregister from the yellow pages
        try {
            DFService.deregister(this);
        }
        catch (FIPAException fe) {
            fe.printStackTrace();
        }
        // Printout a dismissal message
        System.out.println("Supplier-agent "+getAID().getName()+" terminating.");
    }

    /**
     * This is invoked by the GUI when the user adds a new book for sale
     */
    public void updateCatalogue(final String title, final int price) {
        addBehaviour(new OneShotBehaviour() {
            public void action() {
                catalogue.put(title, new Integer(price));
                System.out.println(title+" inserted into catalogue. Price = "+price);
            }
        });
    }
}

```

Appendix

```

/**
 * Inner class OfferRequestsServer.
 * This is the behaviour used by Motor-supplier agents to serve incoming requests
 * for offer from order agents.
 * If the requested book is in the local catalogue the supplier agent replies
 * with a PROPOSE message specifying the price. Otherwise a REFUSE message is
 * sent back.
 */
private class OfferRequestsServer extends CyclicBehaviour {
    public void action() {
        MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.CFP);
        ACLMessage msg = myAgent.receive(mt);
        if (msg != null) {
            // CFP Message received. Process it
            String title = msg.getContent();
            ACLMessage reply = msg.createReply();

            Integer price = (Integer) catalogue.get(title);
            if (price != null) {
                // The requested book is available for sale. Reply with the price
                reply.setPerformative(ACLMessage.PROPOSE);
                reply.setContent(String.valueOf(price.intValue()));
            }
            else {
                // The requested book is NOT available for sale.
                reply.setPerformative(ACLMessage.REFUSE);
                reply.setContent("not-available");
            }
            myAgent.send(reply);
        }
        else {
            block();
        }
    }
} // End of inner class OfferRequestsServer

/**
 * Inner class PurchaseOrdersServer.
 * This is the behaviour used by Motor-supplier agents to serve incoming
 * offer acceptances (i.e. purchase orders) from order agents.
 * The supplier agent removes the purchased book from its catalogue
 * and replies with an INFORM message to notify the buyer that the
 * purchase has been sucesfully completed.
 */
private class PurchaseOrdersServer extends CyclicBehaviour {
    public void action() {
        MessageTemplate mt
MessageTemplate.MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);
        ACLMessage msg = myAgent.receive(mt);
        if (msg != null) {
            // ACCEPT_PROPOSAL Message received. Process it
            String title = msg.getContent();
            ACLMessage reply = msg.createReply();

            Integer price = (Integer) catalogue.remove(title);
            if (price != null) {
                reply.setPerformative(ACLMessage.INFORM);
                System.out.println(title+" sold to agent "+msg.getSender().getName());
            }
        }
    }
}

```

Appendix

```
    }
    else {
        // The requested book has been sold to another buyer in the meanwhile .
        reply.setPerformative(ACLMessage.FAILURE);
        reply.setContent("not-available");
    }
    myAgent.send(reply);
}
else {
    block();
}
}
} // End of inner class OfferRequestsServer
}
```