

# University of Wollongong - Research Online

## Thesis Collection

Title: Multi-agent based modeling and analysis of collaboration strategies in supply chain

Author: Xin Li

Year: 2006

Repository DOI:

### Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following: This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author. Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material.

Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

**Unless otherwise indicated, the views expressed in this thesis are those of the author and do not necessarily represent the views of the University of Wollongong.**

Research Online is the open access repository for the University of Wollongong. For further information contact the UOW Library: [research-pubs@uow.edu.au](mailto:research-pubs@uow.edu.au)

*University of Wollongong Thesis Collections*

*University of Wollongong Thesis Collection*

---

*University of Wollongong*

*Year 2006*

---

# Multi-agent based modeling and analysis of collaboration strategies in supply chain

Xin Li  
University of Wollongong

Li, Xin, Multi-agent based modeling and analysis of collaboration strategies in supply chain, M.Info.Sys.-Res. thesis, School of Economics and Information Systems, University of Wollongong, 2006. <http://ro.uow.edu.au/theses/623>

This paper is posted at Research Online.  
<http://ro.uow.edu.au/theses/623>

## **NOTE**

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

## **UNIVERSITY OF WOLLONGONG**

### **COPYRIGHT WARNING**

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

# **Multi-Agent based Modeling and Analysis of Collaboration Strategies in Supply Chain**

A thesis submitted in partial fulfillment of the requirements for the award of the degree

**Master of Information Systems (by Research)**

From

**University of Wollongong**

By

**Xin Li**

Master of Science (Logistics), University of Wollongong  
Master of Information and Communication Technology, University of Wollongong  
Bachelor of Information Science, FuDan University

**Information Systems  
School of Economics and Information Systems**

**2006**

# **Thesis Certification**

## **CERTIFICATION**

I, Xin Li, declare that this thesis, submitted in partial fulfillment of the requirements for the award of the Degree of Master of Information Systems by Research, in the Information Systems discipline, School of Economics and Information Systems at the University of Wollongong, is wholly my own work otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Xin Li  
October 2006

## **Acknowledgements**

There are several people whose assistance has been invaluable in completing this thesis. I would especially like to thank my supervisor, Dr. Sim Kim Lau, for her help and guidance during the course of this study. I would also like to thank Min He and Chattrakul Sombattheera for their kind help and assistance. In addition, I would like to thank our department, School of Information Systems, has supported me financially to allow me to attend the conferences. Finally I would also like to thank many colleagues of mine, and especially my family who encourage me and support me to continue my study.

October 2006

Xin Li

## List of Publications

This is a list of refereed conference papers related to this research.

Li, X. and Lau, S. 2005, 'A Multi-Agent approach toward Collaborative supply chain management', in *the Proceedings of The 5th International Conference on Electronic Business (ICEB) 2005*, CD-ROM, 5-9 Dec., Sheraton Hotel and Towers, Hong Kong, China.

Li, X. and Lau, S. 2005, 'Collaboration in supply chain: a multi-agent approach', in *the Proceeding of the 6th International We-B (Working For E-Business) Conference (We-B)*, 23rd-25th Nov., Victoria University, Melbourne, Australia.

## List of Figures

		Page No.
Figure 2.1	Supply chain network	24
Figure 2.2	Four basic SCOR Processes in each organizational element	25
Figure 2.3	Four basic SCOR Processes	26
Figure 2.4	Hierarchy of supply chain planning.	28
Figure 2.5	The evolution towards collaborative supply chain planning	32
Figure 2.6	Agents' characteristics - Modularity, Decentralization, Changeability	47
Figure 3.1	Supply chain network	54
Figure 3.2	Framework of a multi-agent based supply chain management system	55
Figure 3.3	Agent-based supply chain coordination	58
Figure 3.4	Function of the distributor coordination agent	59
Figure 4.1	A simple supply chain	67
Figure 4.2	Multi-agent based supply chain system structure	70
Figure 4.3	Supply chain prototype structure and processes	75
Figure 4.4	Supply Chain Simulation Prototype	76
Figure 4.5	Supply chain model in scenario 1 and 2	77
Figure 4.6	Supply chain model in scenarios 3 and 4	78
Figure 4.7	Orders in Scenario 1	79
Figure 4.8	Orders in Scenario 2	80
Figure 4.9	Orders in Scenario 3	80
Figure 4.10	Orders in Scenario 4	81
Figure 4.11	Inventory level in Scenario 1	81
Figure 4.12	Inventory level in Scenario 2	82
Figure 4.13	Inventory level in Scenario 3	82
Figure 4.14	Inventory level in Scenario 4	83
Figure 4.15	Backorders in Scenario 1	83
Figure 4.16	Backorders in Scenario 2	84

Figure 4.17	Backorders in Scenario 3	84
Figure 4.18	Backorders in Scenario 4	85

## List of Tables

		Page No.
Table 2.1	Definition of four management processes in supply chain	26
Table 2.2	The financial and non-financial benefits from effective supply chain collaboration	36
Table 2.3	Superior collaboration can help manufacturers and retailers	37
Table 4.1	Service level and safety factor	73

## List of Abbreviations

AI	Artificial Intelligence
APS	Advanced Planning & Scheduling Systems
B2B	Business-to-Business
CPFR	Collaborative Planning, Forecasting and Replenishment
DSS	Decision Support Systems
EDI	Electronic Data Interchange
ECR	Efficient Consumer Response
ERP	Enterprise Resource Planning Systems
IS	Information Systems
IT	Information Technology
KQML	Knowledge Query and Manipulation Language
3PL	Third Party Logistics
MRP	Material Requirement Planning
MAS	Multi-Agent Systems
SCM	Supply Chain Management
VMI	Vendor Managed Inventory

## Table of Contents

<b>Thesis Certification .....</b>	<b>2</b>
<b>Acknowledgements .....</b>	<b>3</b>
<b>List of Publications .....</b>	<b>4</b>
<b>List of Figures.....</b>	<b>5</b>
<b>List of Tables .....</b>	<b>7</b>
<b>List of Abbreviations .....</b>	<b>8</b>
<b>Table of Contents .....</b>	<b>9</b>
<b>Abstract.....</b>	<b>11</b>
<b>Chapter I Background and Introduction .....</b>	<b>13</b>
1.1 Introduction.....	13
1.2 Research Problems.....	15
1.3 Research Aim.....	19
1.4 Research Objectives.....	19
1.5 Overview of Research.....	19
1.6 Organization of Thesis .....	20
<b>Chapter II Literature Review .....</b>	<b>22</b>
2.1 Supply Chain Management.....	22
2.1.1 Definition of Supply Chains .....	22
2.1.2 Collaboration in Supply Chains .....	31
2.2 Information Technology and Information Systems in the SCM.....	37
2.3 Multi-Agent System.....	43
2.4 Research Gap .....	49
<b>Chapter □ Proposed conceptual framework of multi-agent based supply chain management system .....</b>	<b>53</b>
3.1 Conceptual framework of multi-agent based supply chain management system .	53
3.1.1 Function agents .....	55
3.1.2 Communication agents.....	56
3.1.3 Coordination agents .....	57
3.1.4 Monitoring agents .....	60
3.2 Contributions of the conceptual framework to supply chain collaboration .....	61

3.3 Summary .....	65
<b>Chapter IV A multi-agent supply chain prototype.....</b>	<b>66</b>
4.1 Supply chain structure.....	66
4.2 Overview of the model structure.....	69
4.3 Performance measures and simulation scenarios.....	76
4.4 Simulation results.....	79
4.5 Discussion .....	85
4.6 Summary .....	88
<b>Chapter V Conclusion .....</b>	<b>89</b>
5.1 Research summary .....	89
5.2 Future research direction.....	91
<b>References.....</b>	<b>92</b>
<b>Appendix A - Program Codes.....</b>	<b>97</b>

## **Abstract**

A supply chain is a worldwide network of suppliers, factories, warehouses, distribution centers and retailers through which raw materials are acquired, transformed and delivered to customers. Modern supply chain management is moving away from vertically integrated companies that control all aspects of production and distribution toward networks of independent suppliers and distributors. Nowadays, supply chain collaboration has become the cornerstone of high performance in supply chain management. A key step in this collaboration process is to share information among the supply chain partners. However, current supply chain collaboration mainly focuses on the collaboration between two companies in a supply chain instead of in the whole system due to the limitation of the current modeling method and capabilities of current information systems. The multi-agent approach is a promising modeling method that can be used to design and develop supply chain management system to facilitate supply chain system-wide collaborative management. The aim is to investigate information sharing as a basic supply chain collaboration strategy through the application of the multi-agent approach to model and simulate the supply chain. This research presents a proposed conceptual framework of multi-agent based collaborative supply chain management system. The framework consists of four types of agents that include function, communication, coordination, and monitoring agents. The proposed framework illustrates the application of multi-agent techniques to integrate disparate supply chain information systems, to facilitate information sharing in the supply network, to support collaborative supply chain planning and to coordinate problem solving. A multi-agent based supply

chain prototype is developed to investigate the impacts of information sharing on supply chain performance. Four scenarios have been investigated to measure the performance of both the inventory cost and customer service levels. The simulation results show that information sharing as a basic supply chain collaboration strategy can reduce the bullwhip effect and result in lower amounts of the inventory holding, but it leads to higher stock-outs.

# **Chapter I Background and Introduction**

## **1.1 Introduction**

In today's competitive business environment, industry is recognizing the importance of efficient supply chain management. The supply chain is viewed as a network of facilities that connects from the ultimate suppliers to the ultimate customers; and distribution options that perform the functions of procurement of materials, transformation of these materials into intermediate and finished products and the distribution of the finished products to customers. These autonomous and semiautonomous business entities perform all processes associated with the flows of material and information. Material and information both flows up and down the supply chain. Accompanying the globalisation of business and the increasing outsourcing, the traditional sequential supply chain system has evolved into a complex supply chain network and the competition has transformed from company versus company to supply chain versus supply chain (Iskanian et al. 2004).

The objective of supply chain management (SCM) is to reduce and distribute merchandise at the right quantities, to the right locations, at the right time, in order to minimize system-wide costs while satisfying service level requirements (Simchi-Levi 2003). The focus of the SCM is to optimize the supply chain performance on the whole system level. It not only requires the integration of processes within a single company, but also demands of inter-company collaboration to create synergies in the supply chain.

Supply chain collaboration builds on information sharing, collaborative decision-making and coordinated problem solving. Today, various levels of collaboration techniques based on information sharing were set up in real supply chains with the support of information technology and systems. However, these collaborations only focus on two companies in a supply chain instead of the whole system. This is mainly due to the limited capability of current information systems as well as the complexity and dynamics of the supply chain network (Frey et al. 2004).

A key to meeting the future challenges is the development of next generation information and management systems, which can achieve transparent information flow in business network and support collaborative decision-making and coordinated problem solving at the supply chain system level. A promising approach is the use of multi-agent approach (Iskanian et al. 2004, Lee and Kim 2004). Software agents are regarded as self-interested, autonomous, rational entities having their own objectives and being in charge of a certain sub-task of an overall decision problem. For solving their sub-tasks, agents have to communicate and to coordinate their decisions with other agents. A multi-agent system is one that consists of a number of agents that take specific roles and interact with one another to solve problems that are beyond the capabilities or knowledge of any individual agent (Wooldridge 2002).

This thesis investigates the appropriateness of the multi-agent approach to model the supply chain system in order to facilitate collaborative supply chain management. A conceptual multi-agent based supply chain collaborative management framework is

proposed to demonstrate the application of the agent technology to support supply chain collaboration. Furthermore, a multi-agent supply chain model prototype is developed to analyse the effects of information sharing.

## **1.2 Research Problems**

The key of the SCM is the integration of processes both upstream and downstream in the supply chain. It crystallised concepts about integrated business planning and execution. Information technology (IT) has become an important enabler for this integration process. By optimising and streaming cross-company processes with an information system, the supply chain network can reduce costs, enhance quality and speed up operations. Today, many information systems have been developed for the SCM, from electronic data interchange (EDI) systems and enterprise resource planning (ERP) systems to the newly developed advanced planning and scheduling (APS) systems and e-commerce solutions.

According to Shapiro's decomposition of information technologies (Shapiro 2001), the first two applications of EDI and ERP together with e-commerce solutions belong to the category of "transactional information technology", because these systems are concerned with acquiring, processing and communicating raw data. However, the ready access to transactional data does not automatically lead to better decision-making. As competitive advantage in the SCM is gained not simply through faster and cheaper communication data, another form of IT, known as the "analytical information technology" is required for analysing decisions over short, medium and long term futures. The APS belongs to

this category because it allows analysis of raw data to help managers to plan and make decisions.

Supply chain collaboration builds on information sharing, collaborative planning, and collaborative operation. Information sharing is the first requirement for collaboration. In a supply chain, information is often distributed and controlled by different entities. Information sharing between partners in a supply chain may take place at different levels from transaction processing to strategic information sharing related to operation and planning. Routine transactional information can be exchanged automatically through the EDI and the Internet. However, the high-level information and knowledge sharing required for business coordination may not be easily achieved in the supply chain network among heterogeneous information systems used in different entities in a supply chain (Iskanius 2004).

The APS is a powerful tool for effective supply chain planning. However its capability to support collaborative planning on the supply chain system level is limited. Nowadays, due to the effects of globalization and increasing outsourcing activities, the supply chain consists of many independent entities that are usually separated geographically. Each entity has its own operational strategies, and business decision mechanisms are under different constraints, i.e., they are autonomous in nature. Since the SCM integrates issues of multiple stage production and multiple stage distribution system wide planning, the network is a complicated system. Due to the centralistic view of hierarchical planning and modeling methods underlining today's APS, the APS system cannot support supply

chain collaborative planning on the system level. It may be suitable in an intra-organizational supply chain or a focal inter-organizational supply chain, but not so in such a complex supply chain network because developing an integrated supply chain planning information system may require all systems to be re-developed into a monolithic integrated system capable of handling all foreseeable scenarios for planning and execution. This is regarded as not feasible due to the high development and maintenance costs, not to mention the inflexibility of such a system. Furthermore, the APS is built on a centralized supply chain decision database, therefore if other supply chain partners are reluctant to share their data and to feed them into a central database while insisting on their own planning domain, modeling the SC-wide flows by a single APS is impossible. Therefore, the supply chain collaboration was not sufficiently achieved based on current information systems and modeling method. Effective collaborative supply chain management in today's complex, decentralized, distributed supply chain network requires more effective information technology support.

In a distributed domain such as the supply chain, where any local decision may have widespread effects, a key to meeting future challenges is the development of next generation information and management systems which can support collaborative problem solving and decision making in an integrated supply chain. Recently, multi-agent based systems and technology have been applied as a new paradigm for conceptualising, designing and implementing the software system (Marik and McFarlane 2005). It can be seen as a new technology for improving, or replacing, technologies used in transactional and analytical information technologies. An intelligent agent is software

conceived as an autonomous, integrable and cooperative entity that works across many boundaries that currently separate software systems. Multi-agent systems are distributed systems made of several autonomous agents that take specific roles and interact with one another to solve problems that are beyond the capabilities or knowledge of any individual agent (Wooldridge 2002). These interactions can vary from simple information interchanges, to request for particular actions, and on to cooperation, coordination and negotiation in order to manage interdependent activities (Jennings 2000). Interaction and coordination are the core processes of a multi-agent system (Lee et al. 2004). According to Jennings & Wooldridge (2002), agent technology and multi-agent systems can be used to develop highly complex systems. Paranak et al. (1998, p.24) claimed, "...Agent-based modeling is most appropriate for domains characterized by a high degree of localization and distribution and dominated by discrete decision..." As a supply chain system is a large-scale complex system with decentralization, collaboration and intelligence being its essential characteristics, thus it is reasonable to apply the multi-agent system to model the supply chain network and process, and to implement supply chain management applications using multi-agent technology.

This research proposes the use of multi-agent based technology to model supply chain network and to support collaborative supply chain management. The research will investigate information sharing as a basic supply chain collaboration strategy using the multi-agent approach.

### **1.3 Research Aim**

This research aims to investigate the application of the multi-agent approach to model and simulate the supply chain with special focus on information sharing as a basic supply chain collaboration strategy.

### **1.4 Research Objectives**

The objectives of this research are:

1. To develop a multi-agent based collaborative supply chain management system conceptual framework.
2. To develop a multi-agent based simulation model to describe a supply chain.
3. To analyse the impact of information sharing as a basic supply chain collaboration strategy on supply chain performance.

### **1.5 Overview of Research**

We aim to provide a test-bed for different information sharing strategies based on the multi-agent based supply chain model. We will analyse the impact of demand information sharing on the supply chain performance.

A literature review is conducted to investigate the characteristics of supply chain systems and multi-agent systems. A conceptual framework of agent-based supply chain management systems will be developed. A prototype of a simple supply chain model is

developed using the Java programming language to demonstrate modeling of supply chain using agents approach. Due to time and scope constraints, the prototype is restricted to the ordering and inventory management function of the supply chain. The impact of information sharing using different scenarios will be analysed.

## **1.6 Organization of Thesis**

This thesis is organized into five chapters. Chapters 1 and 2 present the theoretical background related to this research. Chapter 3 presents the proposed conceptual framework of multi-agent based supply chain management system. The design and analysis of the multi-agent supply chain prototype is described in Chapter 4. Finally, Chapter 5 concludes the thesis and future research direction is proposed.

Chapter 1 introduces the context of this research. Research problem is identified. Research aim and objectives are presented, followed by overview of research.

Chapter 2 reviews the literature in the areas of supply chain and multi-agent approach. Process-oriented, integrated, collaborative supply chain management issues will be highlighted, followed by discussion of supply chain information systems. The characteristics of multi-agent systems are also reviewed and analysed in relation to the characteristics of supply chain systems.

Chapter 3 presents the proposed conceptual framework of multi-agent based supply chain management system to facilitate supply chain collaboration. Four types of agents

including function, communication, coordination and monitoring agents are presented. The chapter concludes by discussing the contribution of the proposed framework to supply chain collaboration.

Chapter 4 presents a supply chain simulation prototype developed using the multi-agent approach. This prototype is used to analyse impacts of information sharing. The simulation model is based on a simple supply chain containing four echelons with ordering and inventory management function. The results are analysed based on four scenarios.

Chapter 5 summarises the research results and present future research directions.

## **Chapter II Literature Review**

This chapter presents the literature review in two areas of research: supply chain (S.C.) and multi-agent systems. The chapter is organised as follows. Section one presents concepts of supply chain and some important issues of supply chain management, with specific focus on supply chain collaboration. In section two, the roles of different information systems in supply chain management are discussed. Their limitations in supporting system-wide supply chain collaboration will be identified. The concepts of agent and multi-agent systems are discussed in section three. Following that, examples of applying multi-agent systems in the field of supply chains are outlined.

### **2.1 Supply Chain Management**

#### **2.1.1 Definition of Supply Chains**

There are a number of different definitions of supply chains in the literature. For example: Simchi-Levi et al. (2003, p.1) defines a supply chain as being

*A network of suppliers, manufacturing centers, warehouses, distribution centers, and retail outlets, as well as raw materials, work-in-process inventory, and finished products that flow between the facilities.*

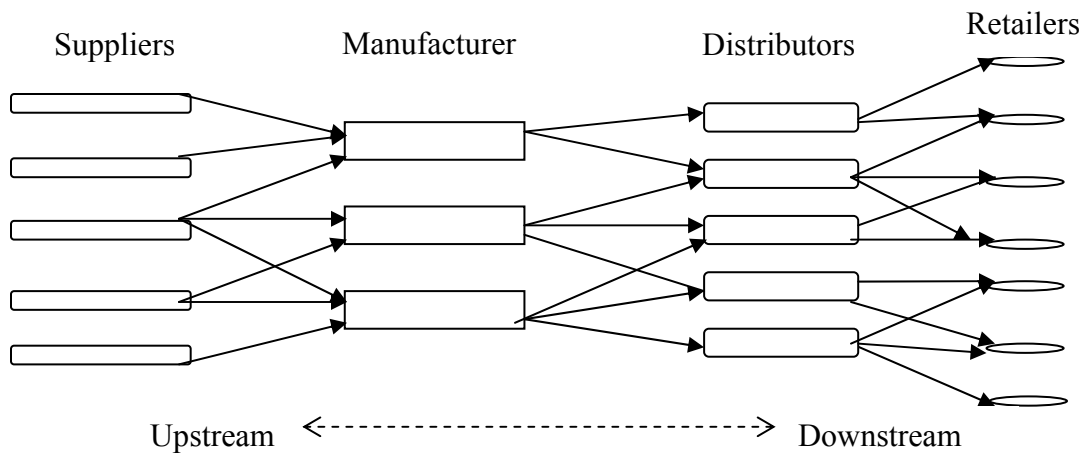
In Christopher's book (1998, p. 15), supply chain is considered as

*A network of organizations that are involved through upstream and downstream linkages, in the different processes and activities that produce value in the form of products and services in the hands of ultimate consumer.*

Chopra and Meindl (2004, p.4) has another analogous definition:

*A supply chain consists of all parties involved directly or indirectly, in fulfilling a customer request. It is dynamic and involves constant flows of information, product and funds between different stages.*

Although the definition of SC is not unique, there is a general agreement in the literature on what a supply chain is. Generally, a supply chain is a set of autonomous or semiautonomous business entities that include suppliers, manufacturers, distributors, wholesalers, and retailers that manufacture and distribute products to consumers. Supply chain activities involve procuring raw materials, converting them into products and distributing them to warehouses and distribution centers. Within the supply chain, inventory buffers are utilised to smooth demand fluctuations thereby reducing the risk of stock-outs to the expense of increased costs. In reality, most supply chains can be structured as networks. A manufacturer may receive material from several suppliers, and supply products to several distributors and customers. Thus, it can be considered as a “supply chain network” as shown in figure 2.1.



**Figure 2.1:** Supply chain network

Figure 2.1 shows a typical supply chain. Materials flow from raw material sources through a manufacturing level, which transforms the raw materials to intermediate produces (also referred to as components or parts). These are assembled on the next level to form products. The products are shipped to distribution centres and from there to retailers and customers. The supply chain shown in figure 2.1 contains four echelons (suppliers, manufacturers, distributors, and retailers), where each level (or echelon) of the chain may comprise numerous facilities. Thus, the complexity of supply chain arises from the number of echelons in the chain and the number of facilities in each echelon. A supply chain can also be regarded as a dynamic process and involves a constant flow of information, material and funds across multiple functional areas, both within and between chain members (Chopra and Meindl 2001). The members in a supply chain perform all processes associated with these flows. The four core management processes in supply chain are: plan, source, make, and deliver.

Concurrent with the increased importance of supply chain to a company's competitiveness has been a shift from traditional function-based (vertical) management to process-based (horizontal) management. As a result, the tight integration of management processes is becoming increasingly important, and complex operation processes must be clearly defined and effectively implemented. The Supply Chain Operations Reference model (SCOR) has been positioned by the Supply-Chain Council (SCC) to become the industry standard for describing and improving operational process effectiveness (Stewart 1997). The framework of the model is based on process description. As illustrated in figures 2.2 and 2.3 (Supply-Chain Council 2000), SCOR is based on four management processes of plan, source, make, and deliver to describe supply chains. This approach allows a supply chain description to be assembled across organizations, internal and external, across industry segments and across geographies. The definitions of these processes are shown in Table 2.1.

**Figure 2.2:** Four basic SCOR Processes in each organizational element (Source: SCOR 2000)

**Figure 2.3:** Four basic SCOR Processes (Source: SCOR 2000)

**Table 2.1:** Definition of four management processes in supply chain (Source: SCOR 2000)

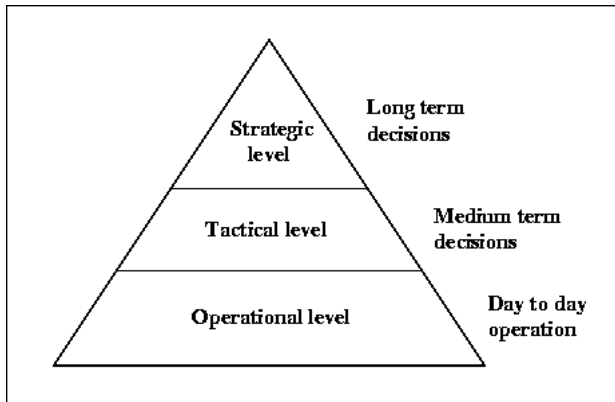
Each organisational element within the chain has “plan”, “source”, “make” and “deliver” activities. “Source” activities are associated with acquiring raw materials and connecting organisations with their suppliers, which include identifying and selecting supply sources, assessing supplier performance; receiving, verifying, and transferring product; and authorising supplier payments and so on. “Deliver” activities are associated with the management of orders and delivery of finished goods, connecting an organisation with its customers. Most organisations have “make” activities that transform raw materials into

finished goods. However, a warehouse or distributor, for example, does not perform “make” activities.

Each SCOR process also can be further described by process type. The three processes of “source”, “make”, “deliver” are of execution type. Execution processes transform or transport materials and/or products, which are triggered by a planned or actual demand that changes the state of material goods. On the contrary, the plan activities, which are associated with decisions concerning production planning, inventory management and vehicle routing, belong to the planning type. It is a process that aligns expected resources to meet expected demand requirements.

The term “management” includes planning, execution and problem solving. According to Simchi-Levi (2003), supply chain management is a set of approaches utilised to efficiently integrate suppliers, manufacturers, warehouses and stores so that merchandise is produced and distributed at the right quantities, to the right locations and at the right time in order to minimize system wide costs while satisfying service level requirements. Successful supply chain management requires many decisions relating to the flow of information, product, and funds. According to the SCOR, supply chain planning includes “supply chain plan”, “sourcing plan”, “production plan”, and “delivery plan”. These decisions fall into three categories or phases, depending on the frequency of each decision and the time frame over which a decision phase has an impact. Supply chain planning is concerned with decisions faced by the firm at these different levels: strategic, tactical, and operational levels (Stadlter and Kilger 2002, p.71; Simchi-Levi et al. 2003; Chopra and Meindl 2004; Shapiro 2001).

Figure 2.4 shows the three levels of supply chain planning and decisions as a pyramid-shaped hierarchy. The decisions on a higher level in the pyramid will set the conditions under which lower level decision are made.



**Figure 2.4:** Hierarchy of supply chain planning.

Strategic planning deals with decisions that have a long-lasting effect on the firm. On this level, a company decides how to structure the supply chain over the next several years. It decides what the chain's configuration will be, how resources will be allocated, and what processes each stage will perform. For example, decision regards the location and capacities of production and warehouse facilities, the products to be manufactured or stored at various locations, the models of transformation to be made available along different shipping legs, and the type of information system to be utilized (Simchi-Levi et al. 2003; Chopra and Meidl 2004). Decisions made on the strategic level are interrelated. For example, decisions on mode of transport are influenced by decisions on geographical placement of plants and warehouses, and inventory deployments are influenced by choice of suppliers and production locations. Supply chain network optimisation modeling

(Gattorna 2003, p. 89) is frequently used for analysing these interrelations and the impact of making strategic level changes in the supply chain.

Tactical supply chain planning is concerned with medium term decisions faced by the firm regarding the refinement and allocation of resources to support integration of functional and geographically dispersed activities. These decisions are typically updated anywhere between once every quarter and once every year. Therefore, the supply chain's configuration determined in the strategic phase is fixed. This configuration establishes constraints within which tactical decisions must be made. Companies start the tactical level planning phase with a forecast for the coming year (or a comparable time frame) of demand in different market. As a result of the tactical planning, companies define a set of operating policies that include purchasing and production, inventory policies and transportation strategies, including the frequency with which customers are visited. These policies will govern short-term operations (Simchi-Levi et al. 2003; Chopra and Meindl 2004).

Operational planning refers to day-to-day decision making such as scheduling, lead-time quotations, routing and truck loading. The time horizon here is weekly or daily, and during this phase companies make decisions regarding individual customer orders. At the operational level, supply chain configuration is fixed and planning policies are already defined. The goal of supply chain operation is to handle incoming customer orders in the best possible manner. During this phase, firms allocate inventory or production to individual orders, set a date at which an order is to be filled, generate pick lists at a

warehouse, allocate an order to a particular shipping mode and shipment, set delivery schedules of trucks and place replenishment orders. Because operational decisions are being made in the short term (minutes, hours, or days), there is less uncertainty about demand information. Given the constraints established by the configuration and planning policies, the goal during the operation phase is to exploit reduction of uncertainty and to optimize performance (Simchi-Levi et al. 2003; Chopra and Meindl 2004). Supply chain decisions can cross many time horizons. According to Shapiro (Shapiro 2003), supply chain management crystallizes concepts about integrated planning, which refers to inter-temporal coordination of supply chain decisions. Furthermore, since the supply chain decisions concerning production planning, inventory planning, sourcing planning and delivery planning are interdependent, these decisions should be taken together.

In a supply chain, the participants belong to different companies and each of them has its own decision rights to make control policies. However, these companies are not isolated; they are impacted on and are impacted by their partners. If the supply chain is operated in a decentralised way, it will not yield improvement for the whole system's performance. For example, in a multi-echelon supply chain, one stage of the supply chain decides to reduce its inventories. In order to maintain its service level, the inventory pressures are often put on its upstream suppliers. If this company's improvement is achieved by only increasing upstream suppliers' inventories, it will hardly yield improvement for the whole system's performance. This deficiency leads to the need of supply chain collaboration. Therefore, effective supply chain management not only requires functional coordination

within the firm but also demands the collaboration between the firms and its suppliers and customers in the supply chain in order to optimize its supply chain.

### **2.1.2 Collaboration in Supply Chains**

According to Quinn (2001), it is believed that whether or not a company can effectively collaborate with the upstream and downstream supply chain participants has become a core business competency. Supply chain collaboration builds on three basic paradigms: sharing information, collaborative planning and coordinated problem solving (Simchi-Levi 2001; Gattorna 2000; Bowersox et. al. 2002). Information sharing is the basic necessity for collaboration in which supply chain partners share information about demand, inventory level and promotional activities. Collaborative planning and problem solving is the advanced content of supply chain collaboration. Collaboration is one step along the supply chain development path for a company. According to Gattrona (2003), the adoption of collaboration tends to be evolutionary. As shown in Figure 2.5, it evolves from integration of internal enterprise operation to the enterprise extension – i.e. collaborative planning with suppliers and customers and then extended to the supply chain system level.

**Figure 2.5** The evolution towards collaborative supply chain planning (Source: Gattorna 2003)

In order to operate efficiently, firstly the company needs to integrate the functions within their own organisations. Internal business functions such as marketing and production must work together to develop mutually accepted demand and supply plans to ensure supply chain activities are agreed and correctly aligned. Once the company has its own operations working effectively it can start to find potential partners for collaborative planning. The easiest place to start is to enter into a collaborative arrangement with its customers and suppliers. Once that is agreed, however, the companies can begin to share supply chain information such as sales forecasts, materials and capacity availability in order to develop a mutually agreed supply chain plan.

The final stage of the evolution towards collaborative supply chain planning is the development of a network of supply chain partners who work together for mutual benefits. This network would encompass the extended supply chain and include outsourced service providers such as contract equipment manufacturers and third-party logistics providers into an intelligent supply chain ecosystem. In general, the customer dominates the supply chain in a form of customer order. The structure of the supply chain has changed from a chain consisting of consecutive actions following each other to a network of ecosystems where different operations parties have a real-time connection with each other.

Today, the development of business and industry towards process-oriented, collaborative management has led us to the situation where it is not individual companies that compete with each other, rather the competition is between rival supply chains (Christopher 1998). Supply chain collaboration has become the cornerstone of high performance in a supply chain. From the evolution towards collaborative planning in the supply chain, the power focus has shifted to the movement of information in the supply chain that becomes increasingly important in the process of managing and controlling. Furthermore, the modern structure of the supply chain highlights simultaneous communication between different parties and integration of the supply chain as a whole.

In practice, there are various levels of collaboration strategies based on information sharing. The most popular forms discussed in the literature are Information Centralisation, Continuous Replenishment program (CRP), Vendor Managed Inventory (VMI), and

Collaborative Planning Forecasting and Replenishment (CPFR) (Simchi-Levi et al. 2003; Bowersox et al. 2002). Information Centralisation is the most basic form of supply chain collaboration in which the retailers broadcast market consumption to the rest of the supply chain. In a CRP strategy, the retailers share real-time inventory data, which is traditionally viewed as sensitive and secret information, with their suppliers. Suppliers use the received data to prepare shipments and continuously replenish retailer inventory at previously agreed-upon intervals to maintain specific levels of inventory (Raghunathan and Yeh 2001). The VMI process is similar to the CRP. It also involves exchange of critical and sensitive information between retailer and supplier, such as retailer's sales history, quantity on-hand, sales volumes, back orders and returns. The difference is that in a VMI system, instead of the retailer itself, the supplier is responsible for creating and maintaining the stock plan for the retailer. The supplier decides on the appropriate inventory levels for each of the products (within previously agreed-upon boundaries) and the appropriate inventory policies to maintain these levels. Therefore, the retailer is free of the need for forecasting and creating the orders as the supplier generates the orders (Kumar and Kumar 2003). In a CRP or VMI strategy, suppliers can gradually decrease inventory levels at the retail store or distribution center as long as the service levels are met. Therefore, inventory levels are continuously improved. However, it is important to mention that although these techniques could be extended to a whole supply chain, current implementations only work between two business partners. The CPFR is developed by the Voluntary Interindustry Commerce Solutions (VICS) Association (VICS Association 2006). It is a standard that enhances VMI and CRP by incorporating joint forecasting and planning. With the CPFR, the participants exchange electronically a

series of written comments and supporting data that include past sales trends, scheduled promotions and forecasts. They concentrate on differences in forecast numbers and attempt to find the cause of the differences and come up with joint and much improved figures (Holmstrim et al. 2002). Like the VMI and CRP, current implementations of the CPFR only include two stages of a supply chain, i.e., retailers and their wholesalers, or distributors and manufacturers.

It is clear that these collaborative activities between firms that integrate processes can provide information visibility across internal functions and organizations, which can reduce the bullwhip effect (Lee et al. 1997; Yu et al. 2001; McCullen and Towill 2002). The bullwhip effect is a phenomenon which states that the variability of consumers' demands gets amplified as it moves up a supply chain and finally become wide fluctuations at suppliers' level (Verdicchio and Colombetti 2002). The bullwhip is a major concern for many manufacturers, distributors and retailers because the increased variability in the order process requires each facility to increase its safety stock in order to maintain a given service level, which leads to increased costs due to overstocking throughout the systems and can lead to an inefficient use of resources. The bullwhip effect significantly decreases supply chain efficiency and increases operation and production costs because it induces a production and inventory level increase.

Collaborative planning can better match supply and demand, reduce inventory risk and increase movement velocity. Collaboration can enable corporations to improve mutual trust and interdependence, and to focus on their own core competencies. Based on

survey results (Mentzer et al. 2000), it has been shown that an effective collaboration in the supply chain can lead to financial and non-financial benefits (see Table 2.2) which, in turn, offers a competitive edge over other supply chains. In another study conducted, it has been shown that large manufacturers and retailers can generally benefit significantly from superior collaboration with downstream supply chain partners (Matchette and Seikel 2004). Table 2.3 summarises the benefits that can be gained by manufacturers and retailers with superior collaboration in the SCM.

**Table 2.2:** The financial and non-financial benefits from effective supply chain collaboration (Source: Mentzer et al. 2000)

**Table 2.3:** Superior collaboration can help manufacturers and retailers (Source: Matchette and Seikel 2004)

Although collaboration strategies adopted by companies add more advantages to their supply chain, these collaboration practices only focus on two stages of a supply chain, usually between manufacturers and distributors, or distributors and retailers. The synchronization stage of collaboration, which focuses on collaboration on the system-wide level, is still some way off for most companies in practice. This is mainly due to the limitation of current modeling method and information technology and systems support.

## **2.2 Information Technology and Information Systems in the SCM**

Information technology, as an important enabler of effective supply chain management, consists of tools used to collect, organize, access, share and analyze information, and act on it to improve performance of the supply chain. According to Simchi-Levi et al. (2003), information technologies pursue four goals:

- *Collect* information on each product from production to delivery or purchase point, and provide complete visibility for all parties involved;
- *Access* any data in the system from a single-point-of contract, e.g., from a PDA linked to the company's information system through a wireless link;
- *Analyse*, plan activities, and make trade-offs based on information from the entire supply chain;
- *Collaborate* with supply chain partners, through risk sharing or information sharing, to achieve global optimization.

Information technology in the supply chain management domain can be classified as “transactional IT” and “analytical IT” (Shapiro 2001). Transactional IT is concerned with acquiring, processing and communicating raw data about the company's past and current supply chain operations, and with the compilation and dissemination of reports summarizing the data. By contrast, analytical IT evaluates supply chain decisions based on models constructed from supply chain decision databases, which are largely, but not wholly, derived from the company's transactional database. Analytical IT comprises of these supply chain decision databases, plus modeling systems and communication networks linking corporate databases to the decision databases. It is concerned with analysing decisions over short, medium and long-term futures. Typical examples of this type of IT are modeling systems for scheduling weekly production, forecasting demand for next month and allocating it to manufacturing facilities, or locating a new distribution center.

To support supply chain management, various information technology and information systems have been developed, such as EDI, early, less sophisticated legacy systems, ERP systems, e-commerce solutions, and the APS systems. According to Shapiro's decomposition of information technologies (Shapiro 2001), the first four applications belong to transactional IT, because they are concerned with acquiring, processing and communicating raw data. On the other hand, the APS systems belong to analytical IT because they allow analyzing raw data to help managers to make supply chain decisions.

The legacy systems evolved as functional solutions using mainframes or minicomputers. In the past, mainframe-based legacy systems were applied to support transactional processes within one specific function such as order entry, inventory control and accounting. These transactional processing systems were isolated and implemented with incompatible hardware and software. These systems lack integration and consistency across functional areas even within a single company (Bowersox et al. 2002; Yuan et al. 2001). Nowadays, in spite of extensive investment in state-of-the art application software, many firms are still dependent on legacy systems for mission-critical applications (Erasala et al. 2003). The problem of integrating with legacy systems was viewed as one of the major challenges and inhibitors to new IT development (Erasala et al. 2003).

The ERP system facilitates the flow of transactional data in a company relating to manufacturing, logistics, finance, sales and human resources. It mainly focuses on providing the visibility of information within the enterprise and the automation of business processes. It offers the promise of homogeneous, transactional data that can

facilitate integration of functional areas within the enterprise at the operational level (Taylor 2004; Simchi-Levi et al. 2003). The ERP mainly focuses on collecting, organizing, accessing, and sharing information within the enterprises, rather than focusing on analyzing data and information to make decisions. Although some ERP vendors have begun to address this issue and add more planning modules, it remains focused on scheduling individual production facilities and not the entire networks of facilities operating in collaboration. The ERP offers two main benefits that do not exist in non-integrated departmental systems: an enterprise-wide view of business encompassing all functions and departments and an enterprise-wide database, in which all business transactions are recorded, processed, accessed, monitored and reported (Umble et al. 2003). Due to the fact that the information processes across functional areas within the enterprise are now integrated, it provides a much better base for inter-enterprise cooperation. However, since the major focuses of the ERP are limited to intra-enterprise operation, these systems lack support for inter-enterprise collaboration in the SCM (Yuan et al. 2001).

Electronic data interchange (EDI), which is one of the more typical supply chain communication technologies, allows companies to place instantaneous, paperless purchase orders with suppliers. It is well known that traditional EDI reduces transaction costs and errors (Mukhopadhyay et al. 1995; Wang and Seidmann 1995; Sirnvasan et al. 1994; Riggins and Mulhopadhyay; 1994). In the last few years, it has been extended to facilitate inter-organizational collaborative processes such as the CRP (Raghunathan and

Yeh 2001; Walton and Gupta 1999; Sanchez and Perez 2003). However the cost to set up EDI systems is relatively high and they are mainly used by large organizations.

The Internet, which provides a low-cost communication infrastructure available almost anywhere in the world, enables information to be gathered and transferred either in real time or on demand. Therefore the Internet offers much more visibility than the EDI. With the rapid growth of the Internet and web technology, e-commerce has been promoted and provides a great potential for networking and interaction between business and consumers (B2C) and between business partners (B2B). E-commerce solutions encompass the electronic buying and selling transactions between organizations, named as e-procurement (Neef 2001; Swaminathan and Tayur 2003). It can be applied to set up e-marketplaces and e-hubs for online bidding and auctions for business competition (Neef 2001; Emiliani 2000; Swaminathan and Tayur 2003). Moreover, e-commerce offers not only the solutions required for inter-company transactions but also the standards that will facilitate connection and communication among corporations (Berger 2003). However, in spite of the fact that e-commerce has improved the supply chain visibility through information sharing and the coordination between buyers and sellers through the Internet, the new technology is still needed to support high-level knowledge sharing (not just transaction data) and to facilitate more robust collaboration (Yuan et al. 2001).

Faster and easier access to transactional data does not automatically lead to better decision-making and a competitive edge in the SCM (Shapiro 2001). The APS system,

such as production scheduling, forecasting, and supply chain network optimization systems, is a powerful tool for effective supply chain planning. It provides analytical applications to optimize the use of supply, manufacturing, distribution, and transportation resources to match the demand. It can be used to tackle strategic, tactical and day-to-day operational problems (Simch-Levi et al. 2003). It uses sophisticated algorithms and relies on inputs of transactional data collected from the legacy or ERP systems (Chopra and Meindl. 2001).

Supply chain collaboration and decentralization is a main challenge to today's APS. Due to the centralistic view of hierarchical planning that underline today's APS, it might be suitable in an intra-organizational SC or a focal inter-organizational SC but not in a complex supply chain network because developing an integrated supply chain planning information system may require all systems to be re-developed into a monolithic integrated system capable of handling all foreseeable scenarios for planning and execution. However this is regarded as not feasible due to the high development and maintenance cost, not to mention the inflexibility of such a system. Furthermore, if partners are reluctant to share their data and to feed it into a central data-base while insisting on their own planning domain, modeling SC-wide flows on a single APS is no longer possible.

From the foregoing discussion it is clear the capabilities of these current information systems to achieve the inter-organizational and system wide coordination and

collaboration are insufficient. Furthermore, these current information systems, which support the components in the supply chain process, are developed by different vendors and are generally disconnected. According to Erasala et al. (2003), typically, these systems have evolved over the years based on various local and company-wide requirements and were rarely integrated. Disparate IT systems and lack of integration and collaboration make it impossible for many corporations to support the required responsiveness and capability demanded by customers (Puckridge and Woolsey 2003). Integrated supply chain management requires more effective information technology support. Intelligent agent technology and multi-agent systems offer the potential to overcome the limitations of current supply chain technologies and offer new means and tools for supply chain collaboration. Agent technology and multi-agent systems will be discussed in the next section of this chapter.

## **2.3 Multi-Agent System**

This section focuses on the second research area addressed in this thesis: multi-agent systems. In this section, the concept of agents will first be defined. Then, the architecture of agents will be discussed, followed by the concept of multi-agent system. Finally, applications of multi-agent systems in different areas including supply chain management will be presented.

Agent-based technology has generated lots of excitement in recent years because of its promise as a new paradigm for conceptualizing, designing, and implementing software

systems (Sycara 1998). Although agents can be understood in many ways, it is quite commonly accepted that agents can be described as software systems that are autonomous, cooperative (social behavior), reactive and pro-active (Wooldridge and Jennings 1995; Jennings et al. 1998; Nwana 1996). *Autonomy* refers to the principle that agents operate without direct intervention of humans or others, have some kind of control over their actions and internal state and act in a manner as to meet its goals on behalf of its user; *reactivity* means agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur; *pro-activeness* means agents do not simply act in response to their environment; they are able to exhibit goal-directed behavior by taking initiative; cooperation with other agents is paramount. In order to cooperate, agents need to possess social ability, i.e., the ability to interact with other agents (and possibly humans) via some kind of agent communication language (Wooldridge and Jennings 1995).

The multi-agent system (MAS) is not a new concept in the area of computer science to solve complex, distributed problems. According to Wooldridge (2002), a multi-agent system is one that consists of a number of agents that take on specific roles and interact with one another to solve problems that are beyond the capabilities or knowledge of any individual agent. These interactions can vary from simple information interchanges, to request for particular actions and on to cooperation (working together towards a common aim), coordination (organizing problem solving activity so that harmful interactions are avoided or beneficial interactions are exploited) and negotiation (coming to an agreement

which is acceptable to all the parties involved) in order to manage interdependent activities (Jennings 2000; Jennings et al. 1998). Interaction and coordination are the core processes of a multi-agent system (Lee et al. 2004). According to Sycara (1998), important characteristics of the MAS are that: each agent has incomplete information or capabilities for solving the problem and thus has a limited viewpoint; there is no global system control; data is decentralized; computation is asynchronous.

Compared with the centralised approach, although MAS are generally less efficient because the distribution restrains optimization, Multi-agent systems have several advantages (Huhns and Stephens 1999). First, multi-agents are easier to understand and to implement when the problem itself is distributed. Paranak et al. (1998, p.24) claimed, "...agent-based modeling is most appropriate for domains characterized by a high degree of localization and distribution and dominated by discrete decision..." This allows the multi-agent system to give more flexibility when taking into account the modularity of the real, modeled system. Second, for some complex and distributed problems, a centralized solution may be impossible, because systems and data are in independent organizations (Sycara 1998). As discussed in the section of supply chain information systems, the centralised approach is not capable of modeling complex supply chains and solving supply chain problems at the system level because of its inability to cope with a high degree of complexity and change. So this is one of the main reasons in favour of multi-agent systems in the supply chain domain. Third, the multi-agent approach allows for the interconnection and interoperation of multiple existing legacy systems. Most industrial agent applications are additions to the existing systems. They need to interface

with legacy systems, many of which are functionally oriented. For example, a shop-floor control system needs to interface with a factory-wide Materials Requirements Planning system that is doing classical scheduling. When integrating the legacy system with the new system, completely rewriting such software tends to be prohibitively expensive and is often simply impossible. Therefore, in the short to medium term, one of the ways that such legacy systems can remain useful is to incorporate them into a wider cooperating agent community in which they can be exploited by other pieces of software. Incorporating legacy systems into an agent society can be done by building an agent wrapper around the software and to encapsulate it as an agent to enable it to interoperate with other systems (Geneserth and Ketchpel 1994; Ciancarini and Wooldridgem 2001). Fourth is reconfigurability. According to Parunak (1998), the two characteristics of agents, modularity and decentralization, combine to make the multi-agent approach support a plug-and-operate approach (shown in Figure 2.6). This enables changing, adding, or removing both hardware and software modules on the fly whenever this is needed owing to equipment failures or as a consequence of a changing plan. The migration from old to new technology can proceed smoothly, without stopping the operation. This also makes systems maintenance significantly cheaper.

**Figure 2.6:** Agents' characteristics - Modularity, Decentralization, Changeability (Source: Parunak 1998, p.6)

Finally, the multi-agent approach is a natural way to modularize complex systems (Jennings 2000; Parunak 1998; Jennings and Wooldridge 1998). It has been pointed out that the flexible, high-level interactions of agents make the design of complex systems easier (Jennings 2000). Complex systems are always distributed, and agent decomposition is very important to manage complexity. Through control decentralization, The MAS provides designers with the means to reduce the complexity of the system control. This can be done by developing a number of functionally specified and modular agents that are specialized at solving a particular problem aspect. This decomposition allows each agent to use the most appropriate paradigm to solve its particular problem. Through coordinating with one another, the agents in the system can work together to solve the interdependent problems.

From the above discussion, we can see that the multi-agent approach has many advantages over the traditional centralized approach. The MAS are suited for problems where a classical centralized solution isn't appropriate, and where the distribution of

information and decision-making is necessary. It can be applied to model complex, distributed systems. The supply chain system is a large-scale complex system. Decentralisation, collaboration and intelligence are its essential characteristics. Thus it is reasonable to apply the multi-agent approach to model the supply chain network and process, and to implement supply chain management applications.

As a contemporary modeling technique for distributed system, the MAS has been used to solve real-work problems in a range of industrial and commercial applications, ranging from manufacturing to process control, air traffic control and information management (Jennings et al. 1998; Marik and McFarlane 2005; Parunak 2000). Recently some agent related supply chain studies have been reported. We now illustrate some of these studies.

Swaminathan et al. (1998) presented a multi-agent approach to model supply chain dynamics. A supply chain library of software components (such as retailers, manufacturers, inventory policy, etc.) has been developed so that customized supply chain models can be built from the library of some specialized agents. Sadeh et al. (2001) presented an agent-based architecture for dynamic supply chain called MASCOT (Multi-Agent Supply Chain Coordinatin Tool). The MASCOT is a re-configurable, multi-level, agent-based architecture for coordinated supply chain. Agents in the MASCOT serve as wrappers for planning and scheduling modules. The DASCh was developed by Parunak and VanderBok (1998) and Baumgaertel et al. (2001) to explore the modeling techniques of networks of suppliers' suppliers. In particular, flows of products and information flows are viewed as agents to model imperfections in these flows. Agent Building Shell at the

University of Toronto is a library of software classes providing reusable elements for building agent systems. These agents have four layers: a layer for knowledge management, an ontology layer, a layer of cooperation and conflict solving, and a layer of communication and coordination. This latter layer is insured by COOrdination language (COOL).

## **2.4 Research Gap**

In this chapter, research on supply chain management, information systems in supply chain management and multi-agent systems have been reviewed. From the foregoing discussion in this chapter, a major trend of supply chain management is supply chain collaboration, which builds on information sharing, collaborative decision-making and problem solving. In the business world, many implementations of supply chain collaboration have been conducted. However, these implementations usually only include two echelons of supply chain, and not the entire supply chain system.

From the literature review, we identified that it is difficult to achieve information sharing in the whole supply chain system using current approaches. One of the reasons is that in a complex and dynamic supply chain, many companies may not want to disclose their information to another company. Also, even for companies who are willing to share their information, incompatibility among heterogeneous information systems can hinder information sharing. Therefore, global information sharing between companies in supply chains is not always possible, and it is difficult to effectively utilize shared information based on current approaches.

Secondly, conventional centralised and hierarchical approaches applied to current planning systems in supply chain management are inadequate, especially under conditions of disruption and long-term change. The centralised approach under today's APS systems can fail because of its inability to cope with a high degree of complexity and change.

Thirdly, the current information systems, which support the components in the supply chain process, are developed by different vendors and are generally disconnected. According to Erasala et al. (2003), typically, these systems have evolved over the years based on various local and company-wide requirements and were rarely integrated. Disparate IT systems and lack of integration and collaboration make it impossible for many corporations to support the required responsiveness and capability demanded by customers (Puckridge and Woolsey 2003).

Therefore, successful supply chain management requires deploying more sophisticated and advanced approaches that support inter-enterprise and system wide distributed information exchanging activities, decision making, and plan revision to achieve the supply chain integration and system wide collaboration. Intelligent agent technology and multi-agent systems have shown to be applicable to overcome many limitations of current supply chain technologies and offers new means and tools for supply chain collaboration (Iiskanius et al. 2004; Xue et al. 2005; Gutpa et al. 2001; Frey et al. 2003). We have also reviewed a few projects and literatures regarding applying the multi-agent approach in

supply chain management. These studies demonstrate the advantages of multi-agent technology for modeling supply chain components and interactions between the components via specific communications protocols for negotiation and contracts. However, we feel that not enough effort has been put into clearly mapping the supply chain processes to the multi-agent systems, and there is a lack of framework to adopt the multi-agent approach to support supply chain collaboration. Therefore, in this thesis, we investigate a conceptual framework of multi-agent based collaborative supply chain management and propose a conceptual framework of multi-agent approach to synchronize information flow and decision making in supply chain network.

## **2.5 Summary**

In this chapter, we have reviewed the supply chains, supply chain information systems and multi-agent systems. A supply chain is defined as a network of suppliers, factories, warehouses, distributor centers and retailers through which raw materials are acquired, transformed into products that are then delivered to customers. This type of network, in general, involves heterogeneous environments. Supply chain management aims to optimize its performance on the system level. To realise this objective, SCM demands process integration within a company and the collaboration of planning and actions among other supply chain partners. Various supply chain information systems have been developed to facilitate the SCM and to enable process integration. However, current information systems only support collaboration between two levels of a supply chain and not in the system-wide level. System-wide coordination and collaboration, which are not enabled sufficiently by traditional information systems, require more effective modeling

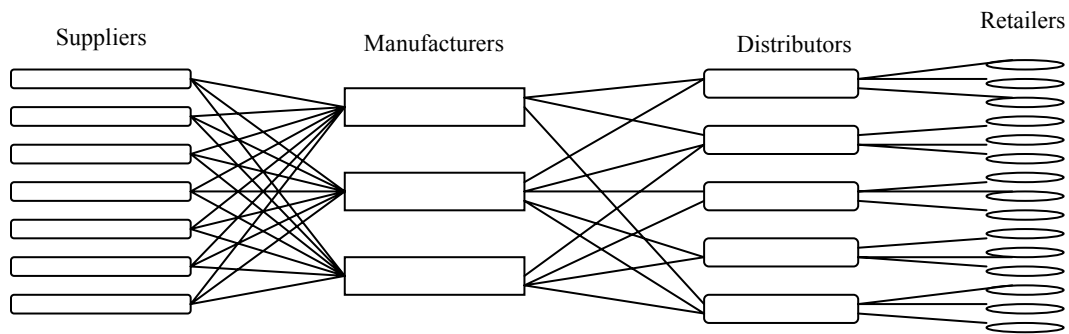
methods and information technology support. Intelligent agent technology and multi-agent systems have shown great potential in overcoming many limitations of current supply chain technologies and supporting collaboration in supply chain management, particularly in supporting transparency in information flows and modeling of the dynamic supply chain for collaborative supply chain planning. In the next chapter, a proposed framework of a multi-agent based collaborative supply chain management system will be presented.

## **Chapter □ Proposed conceptual framework of a multi-agent based supply chain management system**

In chapter two, we have reviewed the trend of supply chain collaboration and have investigated the appropriateness of applying multi-agent techniques to model the supply chain systems. In this chapter, a conceptual framework of a multi-agent based supply chain system to support supply chain collaboration is proposed. This framework illustrates the application of multi-agent techniques to integrate disparate supply chain information systems, to facilitate information sharing in the supply network, to support collaborative supply chain planning and to coordinate problem solving. Finally, we will discuss the contribution of the proposed framework to supply chain collaboration.

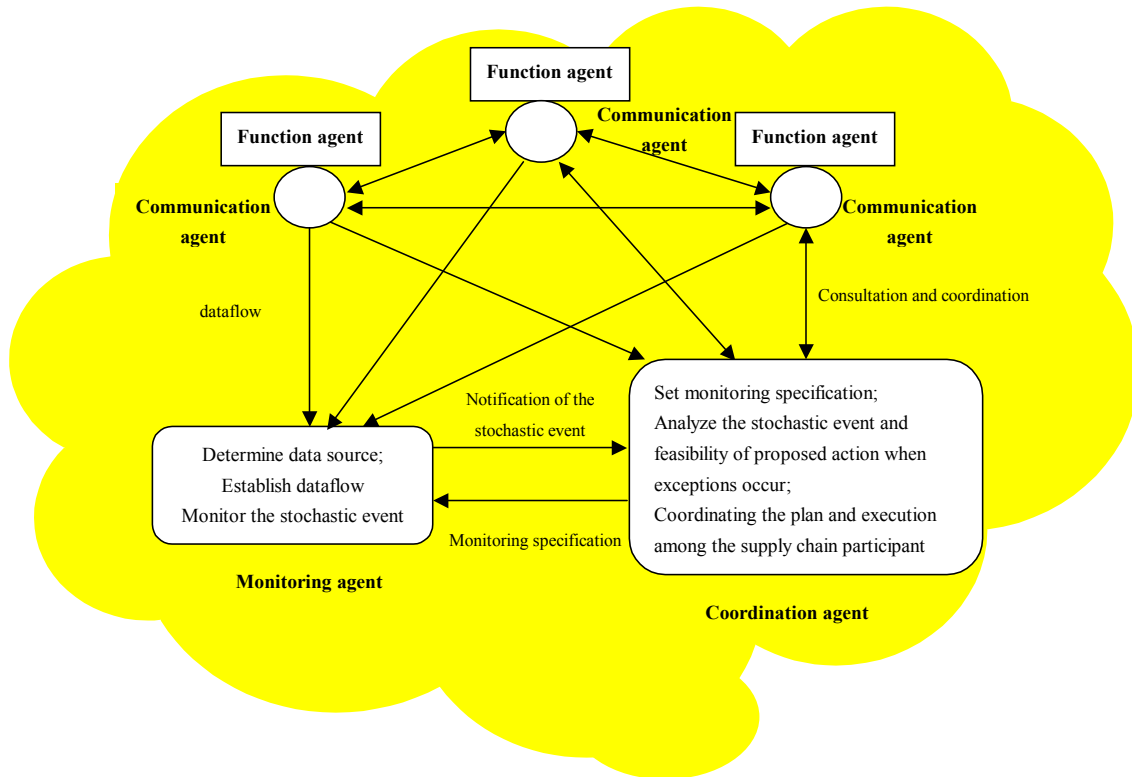
### **3.1 Conceptual framework of a multi-agent based supply chain management system**

This section presents a multi-agent systems framework for supply chain collaboration. The generalized supply chain model is presented in figure 3.1, which is assumed to be a worldwide network consisting of several manufacturers, distributors, suppliers and logistics service providers.



**Figure 3.1:** Supply chain network

Figure 3.2 presents the proposed framework of a multi-agent based supply chain system. There are four types of agents: function agents, communication agents, coordination agents and monitoring agents. Our proposed framework is considered as an “add-on” or “wrap-into” for the current existing system. It will interface with current legacy systems, which are functionally oriented, such as a shop-floor control system interface with a factory-wide MRP system. This way, the legacy systems are encapsulated as the function agent and are connected to the new system. Coordination, communication and monitoring agents are designed to accomplish the procuring, manufacturing and distributing tasks in the supply chain together with the function agents. We will discuss the roles and functions of each of these agents in the proposed supply chain management system.



**Figure 3.2:** Framework of a multi-agent based supply chain management system

### 3.1.1 Function agents

In a supply chain system, the supply chain processes include activities such as planning, manufacturing, assembling, distribution, ordering, and transportation etc. These processes, which belong to different entities in the supply chain, are modeled as function agents. These function agents perform particular tasks in the system and aims to achieve local optimisation. There are existing and separate management information systems that are being used by the various suppliers, distributors, manufacturers and logistics service providers. These management information systems are usually utilised in different parts of the planning, scheduling and execution processes such as capacity analysis, ERP, manufacturing execution system, forecasting software and so on. To completely rewrite

such systems to interconnect them into the newly distributed supply chain system framework would be prohibitively expensive. Therefore, in the proposed framework, these existing separate systems in different organizations are encapsulated as function agents by building agent wrappers around them (Genesereth and Ketchpel 1994). This way, these legacy systems are incorporated into a wider cooperating agent community. Through communication agents, supply chain partners can communicate seamlessly in the framework and through coordination agents, these function agents interact and coordinate with other agents to facilitate system-wide optimization. For example, a forecasting software can be wrapped into a forecasting agent, using some specific algorithms to predict future customer demands. The inventory and warehouse management systems can be wrapped into an inventory control function agent to handle inventory management and stock replenishment; procurement systems software can be wrapped into an order agent to be responsible for acquiring orders from customers and handling customer requests for order modification or cancellation. Other examples include using transportation agents to model the transportation management function, which is responsible for assignment and scheduling of transportation resources to satisfy requests of goods movement; manufacturing agents to model the production planning and scheduling processes.

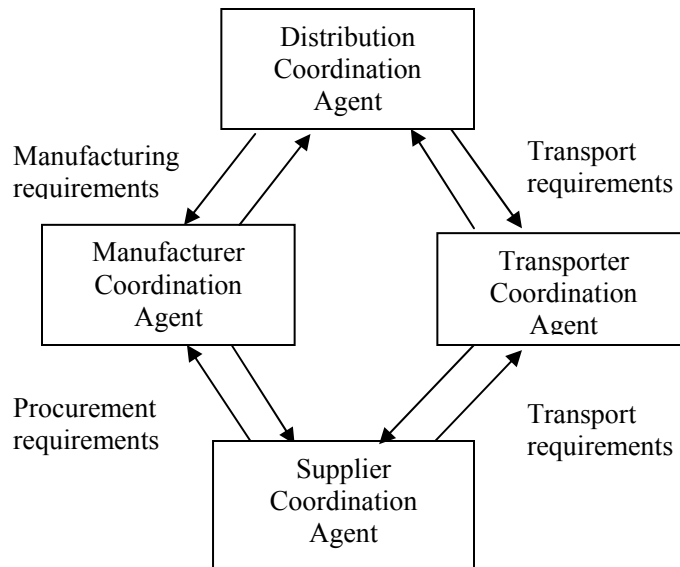
### **3.1.2 Communication agents**

Cooperation and coordination of agents in a multi-agent system requires agents to be able to understand each other and to communicate with each other effectively. Communication agents can be used as a communication interface across a variety of

boundaries to enable the monitoring agent, function agent and coordination agent to communicate with each other. For instance, communication agents can be designed with a translation function to be used to overcome language boundaries within global supply chains. As the supply chain functions usually belong to different organizations, the communication agent designed with authorization function can be used to overcome functional and organizational boundaries. Furthermore, the existing supply chain information systems are encapsulated as function agents and can interact through the communication agents, which work as an interface to overcome the system boundaries. In this way, through function agents and communication agents, the disconnected information systems can be integrated to facilitate supply chain collaboration and coordination.

### **3.1.3 Coordination agents**

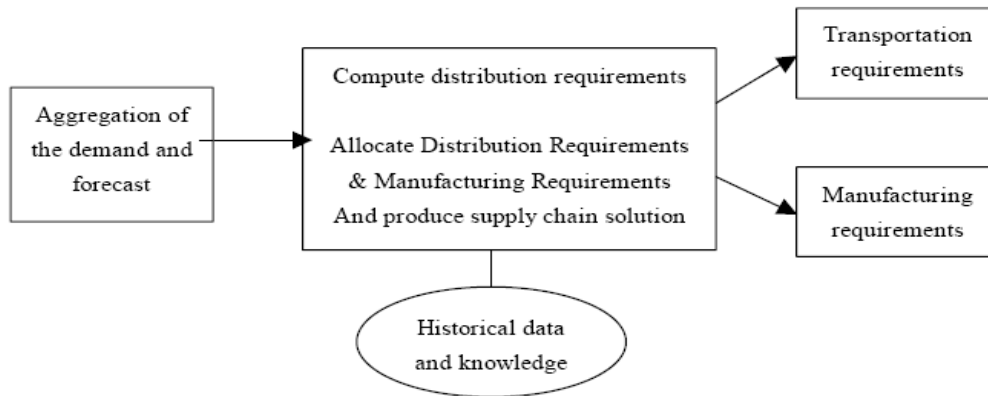
To optimise supply chain decisions, an agent cannot just solely make local optima decisions, it also need to determine what effect those decisions would have on other agents. Ideally, it should coordinate with other agents to find the best alternative that is optimal over the entire supply chain system. In the proposed framework, the coordination among various supply chain functions is accomplished through a group of coordination agents. These coordination agents are designed with specific expertise, knowledge and coordination mechanism to coordinate activities among agents. The coordination agents are further classified into distribution coordination agents, manufacturer coordination agents, transporter coordination agents and supplier coordination agents (as shown in figure 3.3).



**Figure 3.3:** Agent-based supply chain coordination

- a) Distribution coordination agent:** the distribution coordination agent aggregates data and information from the distributors in different areas, performs distribution requirements computation and can establish the distribution requirement plan. Figure 3.4 shows the functions of the distributor coordination agent. The distributor coordination agent will gather or obtain information such as type, quantity, price, and due date from other components of the supply chain to establish a distribution plan. The “type” information refers to the various kinds of products for manufacturing and transportation. The “price” information is the summation of all the prices of various products ordered by retailers. The “due date” information refers to the final date when the different distributors can receive the products, and the “quantity” information is the desired quantity of each type of product. In addition, the distribution coordination agent will determine the type of products to be produced by the specific manufacturer,

or by several manufacturers according to historical data and knowledge. Then, the distribution coordination agent transfers the due date of the type of products into total lead-time for manufacturers and for transporters. Furthermore, the distribution coordinator agent can coordinate with the transporters agent and the manufacturers to reach a feasible and optimized global supply chain solution.



**Figure 3.4:** Function of the distributor coordination agent

- b) Transporter coordination agent:** The function of transporters is to offer pickup and delivery service from different manufacturers to destination terminals or distributors. The operation decisions facing a transporter agent include: to select the right carriers based on costs and lead-time; to consolidate consignment for economics of scale to minimize total costs and maximize on-time delivery. The transporter agent receives the transportation requirement from the distributor coordination agent and performs local optimization using its data on routes, schedule and consignment to conduct candidate transportation commitments.
- c) Manufacturer coordination agent:** The main function of the manufacturer agent is to select manufacturers or production plants, which can produce the required type and quantity within the required lead-time with minimized cost. The minimization of the

total cost is subject to a set of constraints, such as quantity constraints, capacity constraints, customer service level and lead-time. The manufacturer agent receives manufacturing requirements from the distribution coordination agent and performs local optimization to find feasible manufacturing solutions (specified with type, quantity, lead time and price) based on capacity, inventory, production capacity and process time. It provides solutions to the distribution coordination agent for selection.

- d) Supplier coordination agent:** The supplier coordination agent is responsible for selecting suppliers that can minimize costs and maximize prompt material availability. This agent also generates purchase orders for goods. It receives procuring requirements, estimates the resource demand and determines purchasing quantities and timing. It is also responsible for selecting suppliers that can minimize costs and maximize prompt material availability.

#### **3.1.4 Monitoring agents**

The monitoring agent receives the monitoring criteria for disturbance events such as processing rates, and notifies the coordination agents when such events occur. The coordination agents decide what constitutes an exception to a stochastic event. It evaluates how much the occurrence of one stochastic event in the specific supply chain process will influence the other supply chain processes and the whole supply chain process. Then it will identify the monitoring specification of the stochastic events in the different supply chain processes. In addition, the monitoring agent determines what data is to be monitored to detect such exceptions according to the monitoring specification identified by the coordination agent and then conducts the monitoring of data and

information flow. When notified by the monitoring agent of the occurrence of an exception event, such as when an important material is overdue, the coordination agent will analyze the severity of the event, coordinate with other agents to reschedule, reallocate and make appropriate decisions in consultation with the function agent. Human decision-makers may also be involved in this process.

### **3.2 Contributions of the conceptual framework to supply chain collaboration**

One of the major advantages of the proposed multi-agent framework is the ease that it lends to the conceptualization of the supply chain system, which can be visualized as a set of entities and processes. Entities may be suppliers, manufacturers, distribution centers, retailers, customers, or it may be internal departments such as sales, planning, purchasing, materials etc. An entity is responsible for a set of processes, e.g. sales might be responsible for processes related to order acquisition, procurement for processes related to supplier selection and material ordering. In the proposed framework, these different processes in different organizations are modeled as autonomous function agents. There is only a relatively small step from describing a supply chain system to designing it as a multi-agent system, reducing the errors in the translation process. As discussed in chapter two, a supply chain is a domain, which is frequently subject to structural changes. The different processes in the supply chain are modeled as function agents, which are autonomous and distributed. This structure gives a robust system that can undergo continuous adaptation to the changes in the environment, both locally and globally.

Furthermore, in the proposed framework, each function agent performs its functions asynchronously as required and interacts with agents from other companies with the help of communication agents via the Internet. The communication agent, which works in the Internet environment, has autonomous, proactive characteristics. It can autonomously discover the situation and deliver the information to the right parties (function agents or other agents) according to its knowledge. Furthermore, it can be proactive, actively working with other agents to perform actions. This characteristic allows the communication agent to anticipate the needs of the function agent or user. The communication agent may even take initiative in interrupting the function agent or user if a higher-priority event occurs.

As discussed in chapter two, supply chain collaboration builds on information sharing, collaborative planning, and coordinated problem solving. Information sharing is the first requirement for collaboration. In the proposed framework, information is distributed, recorded and controlled by different function agents in different organizations. When building the system, the function agents are given the responsibility of recording data, which pertain to the agent's area of operation. For example, a sales agent's responsibility is to record data on incoming orders, a distribution or transport agent could be responsible for data on deliveries to customer, a procurement agent is obligated to keep statistical data on supplier deliveries, and so on. In this manner, data is recorded locally. It can easily be made accessible throughout the supply chain by letting the interested agents query the local function agent with the assistance of communication agents. For example, when the customers place orders, they want to know when their products will

be delivered. In the proposed system framework, the function agent, a sales agent, will be able to query other agents for the necessary information such as production, inventory level, and current backlogs. Then correct lead times can be deducted from the acquired information.

As discussed in chapter two, global optimization and collaborative planning are important to improve competitiveness from the entire supply chain perspective. However, conventional centralized and hierarchical approaches applied to the current planning systems in supply chain management are inadequate to cope with the high degree of complexity and change. The proposed multi-agent framework can fill this gap. In the proposed framework, individual company planning functions are modeled as function agents, which aim to achieve local optimization. The coordination agents which are composed of procurement coordination agents, distribution coordination agents, manufacturing coordination agents, and transporter coordination agents, are proposed to coordinate among various supply chain function agents. These coordination agents are autonomous, and have social ability and a level of reactivity. These characteristics of the coordination agents make them achieve a high level of coordination. It can dynamically collect data from individual function agents, which belong to different companies, analyze the situation, cooperate and negotiate with other agents and suggest the course of action to all related agents with the help of communication agents. With this approach, the planning and coordination of procurement, transportation and manufacturing can be internal as well as external, providing more flexibility in line with the philosophy of a collaborative supply chain system. For example, in a supply chain there are several

manufacturing plants in the chain. In every manufacturing plant there is a function agent, which is responsible for planning production for that site in the supply chain but information can be passed to and received from manufacturer coordination agents. The manufacturer coordination agent can interact and negotiate with the planning function agents from different manufacturers to obtain a plan that is optimal not only locally, but also globally for the supply chain as a whole. This way, collaborative planning can be achieved through the proposed approach.

Since the supply chain operation is very complex, there will always be uncertainties, exceptions and problems encountered. The smooth operation of a supply chain requires real-time monitoring and cooperation between supply chain partners in an ever-changing environment. The proposed multi-agent framework would allow a high degree of reactivity to unforeseen events through the autonomous, proactive and reactive characteristics of the monitoring agent. The monitoring agent is designed to detect the problem, collect information such as where the problem is and how severe it is and deliver the warning information to the affected entities for appropriate action. To make coordinative problem solving action such as rescheduling the production plan due to the breakdown of a production machine, the coordination agents negotiate with each other and the function agents within the company or with other companies to resolve conflict, and to make appropriate adjustments through real time scheduling. In this way, supply chain event management can be achieved.

### **3.3 Summary**

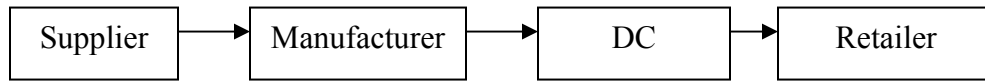
In this chapter, a framework of multi-agent system that is capable of supporting collaborative supply chain management is presented. With this approach, a set of agents with specialized expertise has been designed. The communication agents, which can overcome various language barriers and system barriers, will facilitate the communication among agents and the information distribution and retrieval in the system. In this way, it can facilitate the information sharing among the supply chain participants and improve the visibility of the supply chain network. The proposed framework encapsulates the current existing systems, such as the legacy system, ERP, and APS, into the function agents and connects these function agents with coordination agents through certain coordination mechanisms such as negotiation to reach system optimization. The monitoring agents monitor the real-time supply chain performance and inform related agents and decision makers when an exception event occurs. The related agents and coordination agents work together to amend the plan and deal with the exception event.

## **Chapter IV A multi-agent supply chain prototype**

In this chapter the multi-agent modeling techniques are applied to model and simulate a simple demand-driven supply chain system. The prototype is used to study the value of information sharing, which is one of the central issues in supply chain management and is the basic paradigm of supply chain collaboration, as discussed in chapter two. In the prototype, function agents and communication agents are designed to model the supply chain structure and processes. The chapter is organised as follows. Firstly, the supply chain structure and process are described. Secondly, a multi-agent model is presented that is followed by the discussion on system performance measures and simulation scenarios. Finally, the simulation results are presented together with discussion of the impacts of information sharing on supply chain performance.

### **4.1 Supply chain structure**

In developing the prototype, we have considered a four-stage supply chain that consists of a raw materials supplier, a manufacturer, a distribution centre and a retailer. These entities are connected as shown in Figure 4.1. Each entity has its own set of attributes. The attributes for each entity include: historical demand data, future demand data, forecast demand data, scheduled receipt, backorder, initial inventory level, inventory level, delivery lead-time, information exchange mechanism, forecasting technique, inventory policy, safety-stock level, and service level.



**Figure 4.1:** A simple supply chain

In the prototype, we assume that the supply chain processes a simple product. Each entity, except the supplier, purchases the product from the upstream entity, and sells it to the downstream entity. The last downstream entity, the retailer, sells the product to the end-consumer. The retailer receives the demand from the end-consumer and has accessed to historical demand data of the end-consumer. Each stage has to satisfy the demand of its immediate downstream entity. It has access to the historical demand data of its downstream entity, and assumes that they may or may not receive information about the demand of end-consumer from its downstream entity depending on experimental scenarios, which will be described later. In addition, each entity can receive information on backorders and delivery lead-time from the upstream entity.

At the start of each period, an entity receives its orders from the downstream entity and the scheduled receipts from the upstream entity. Then the entity processes the order and delivers the products to its downstream entity. The supplied products will be received by the downstream entity after a certain time delay, and it is assumed that these replenishment lead-times are known and constant. After that, the entity forecasts future demand based on historical downstream orders or real end-consumer orders, depending on the scenarios of the simulation that are to be discussed later in this chapter.

The safety inventory is adopted in this model to cover the uncertainties on demand. The calculation of the safety inventory level is based on the replenishment lead-time and the coefficient of variation of the observed demand. With this information and the inventory management policy, the expected inventory requirement for the planning horizon is computed. Next, the entity reviews its inventory level and places an order on the upstream stage to replenish its inventory. The order sent by the entity becomes the demand for the upstream entity. It is assumed that there is no uncertainty and delay in the movement of information between supply chain entities, which means the upstream entities process the order immediately after they receive it from their customers. This order is matched against the inventory level of the upstream entity. The entity, which received the order, needs to deliver as many products to its downstream as it can. Any orders greater than the inventories at the upstream entity that are unfulfilled are backordered. The backorder quantity of an entity is the future scheduled receipts, and the order placed in the current period is delivered after replenishment lead-time periods.

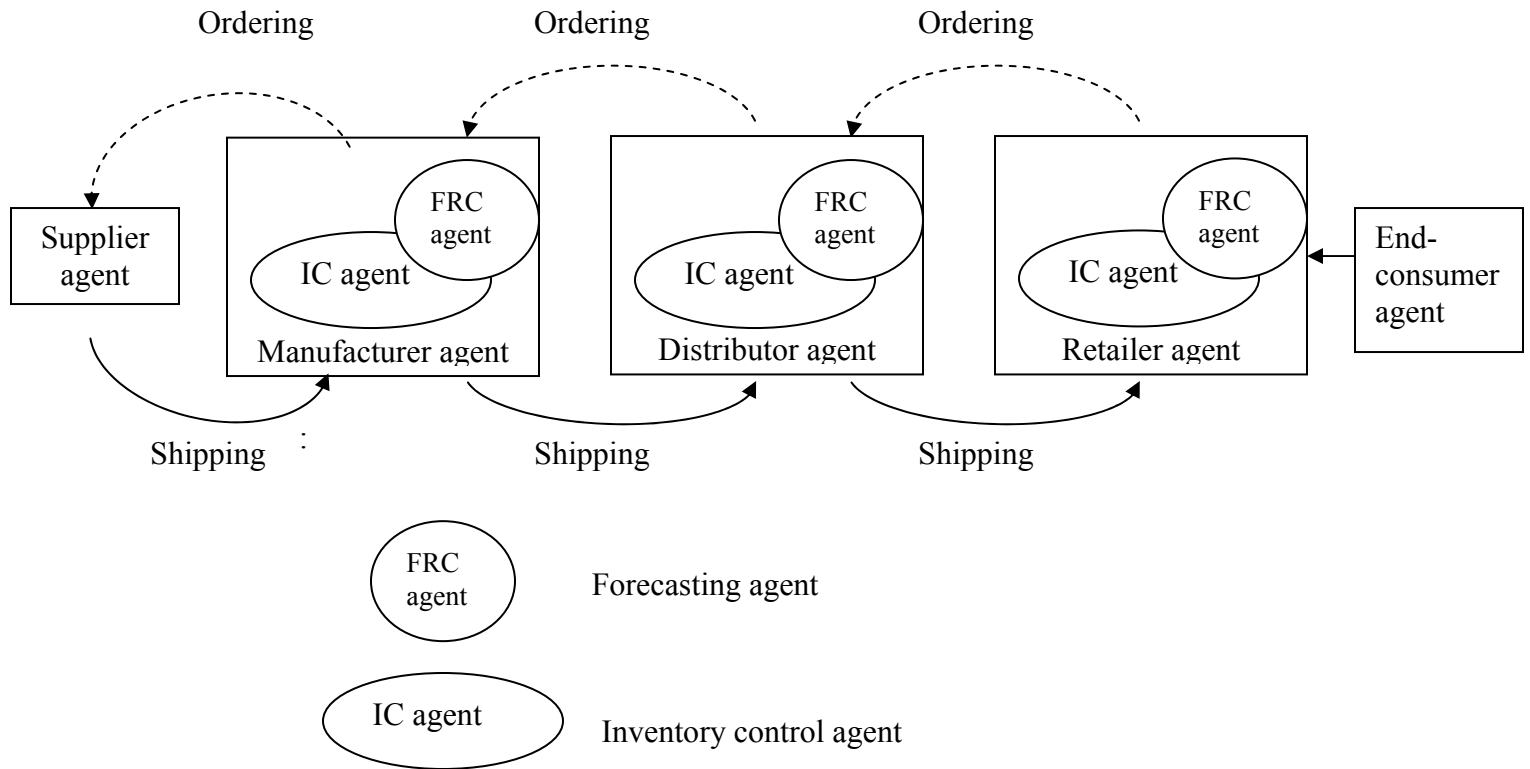
In summary, within each period, each partner in the supply chain will perform the following processes:

1. Receiving orders from downstream
2. Receiving delivered products from upstream
3. Forecasting the future demand and computing order sizes
4. Submitting replenishment orders to upstream entity.

The cycle of this supply chain process moves upstream beginning with the retailer and culminating at the supplier.

## **4.2 Overview of the model structure**

We have adopted the multi-agent technique to model this four-stage supply chain. The purpose of the proposed multi-agent prototype is to study the value of information sharing. Information sharing is one of the central issues in supply chain management and is the basic paradigm of supply chain collaboration. The prototype aims to model and evaluate various information exchange mechanisms. The prototype is developed using Java. Due to the scope of study and time constraints, we can only focus on modeling the ordering and inventory management functions of the supply chain. The prototype includes two types of agents: communication agents and function agents, which have been described in chapter 3. Six function agents - three forecasting agents and three inventory control agents are developed to conduct the forecasting and inventory management functions in the supply chain. There are four communication agents: retailer, distributor, manufacturer, and supplier agents. Each agent, except the supplier agent, is connected to a forecasting agent and an inventory control agent, as shown in Figure 4.2.



**Figure 4.2:** Multi-agent based supply chain system structure

The company agents represent different firms that trade with one another in a supply network. They not only facilitate the communication and information sharing among agents within or among supply chain partners, but also encapsulate the functions of the ordering and goods delivery. In each period, the retailer, distributor, and manufacturer agents observe their downstream customer demands and place orders to their upstream company agent. They also consume inputs from their upstream suppliers and transform them into outputs that they send to their downstream agent. Generally, there are three types of inventories in the manufacturing site: raw product inventory (RPI), work in process inventory (WIP), and finished goods inventory (FGI). This prototype only considers the raw product inventory and the finished goods inventory. We assume that

the manufacturer has unlimited production capacity and the finished product is assembled from one type of raw material. The supplier agent is modeled as a special delivery agent with adequate materials.

To determine how much to order from the upstream company agent, the retailer, distributor, and manufacturer agent need to forecast their customer demand. The forecasting agents are used by the company agents to forecast future demand, and the forecasting techniques are based either on the observed downstream customer demand data or on end-consumer demand data. The most common forecasting techniques are last period demand, arithmetic average, moving average, regression analysis and exponential smoothing (Tersine 1994). The forecasting agents support the exponential smoothing forecasting to forecast future demand. In the exponential smoothing forecast, the forecast average demand period is a weighted average of all previous demand observations, where the weight placed on each observation decreases with the age of the observation (Hax and Candea 1984). In this case, the entity in the supply chain estimates the mean lead-time demand as  $(L+1)\tilde{u}_t$  with

$$\tilde{u}_t = \alpha D_{t-1} + (1-\alpha) \tilde{u}_{t-1}, \quad 0 < \alpha \leq 1 \quad (4.1)$$

where  $\tilde{u}_t$  is the estimated customer demand in time  $t$ ;  $L$  is the lead time (where the lead time is the time from an order is sent from a customer to a supplier until the products ordered arrive at the customer);  $D_{t-1}$  is the observed customer demand in time  $t-1$ ;  $\alpha$  is the smoothing constant and in general,  $0 < \alpha \leq 0.30$ . In other words, the current forecast,  $\tilde{u}_t$ , is the weighted average of the previous period's demand and the previous period's forecast demand, and  $\alpha$  is the relative weight to be placed on the current period's demand.

We assume that the company agents (retailer, distributor, and manufacturer agents) estimate the standard deviation of the forecast errors using sample standard deviation of the single period forecast errors as follows:

$$\sigma_t = D_t - \tilde{u}_t. \quad (4.2)$$

After the company agents (except the end-consumer and the supplier) estimate future downstream customer demands, they contact their inventory control agents to review their inventory position and determine the replenishment order size. The inventory control agents are the third type of agents in this prototype. These agents model the inventory control policy and algorithms used by the company to determine their optimized or desired inventory levels, to estimate when inventory is in danger of falling below specific levels, and to place orders to replenish inventory early enough to allow for estimated delivery times of suppliers.

There are two kinds of inventory control policy: continuous review policy and periodic review policy. In the continuous review inventory control policy, inventory is reviewed every day and a decision is made about whether and how much to order. In many real-life situations, the inventory level is reviewed periodically, at regular intervals rather than continuously, and an appropriate quantity is ordered after each review. This inventory control policy is called periodic review policy. In this prototype, we assume the supply chain participants follow the periodic review inventory control policy, in which the participants review inventory at the end of every review period and calculate the order size and place an order at that time.

There are several approaches available to determine optimum order size when the demand is stochastic such as lot-for-lot ordering, Wagner-Whitin algorithm, Silver-Meal heuristic, periodic order quantity, and part-period algorithms (Tersine 1994). In this prototype, a simple order-up-to inventory method is used in each stage of the supply chain to raise its inventory level up to a given level in each period. One common form of this method is to set the target inventory level in period  $t$ ,  $y_t$ , equal to

$$y_t = (L+r)\tilde{u}_t + z\sqrt{(L+r)} \tilde{\sigma}_t \quad (4.3)$$

where  $r$  is the review period,  $(L+r)\tilde{u}_t$  is an estimate of the mean lead time demand ( $u$ ), which cover demand during a period of  $(L+r)$  weeks,  $\sqrt{(L+r)} \tilde{\sigma}_t$  is an estimate of the standard deviation of the forecast errors ( $\sigma$ ) over the lead time. The target inventory level calculated by formula (4.3) has two components: estimate demand during an interval of  $(r+L)$  weeks and the safety stock, which is the amount of inventory that the warehouse needs to keep to protect against deviations from estimated demand during a period of  $(r+L)$  weeks. The quantity is  $(z\sqrt{(L+r)} \tilde{\sigma}_t)$  and its parameter  $z$  is chosen to meet a desired service level. We assume the service level is 97%, so  $z = 1.88$ . Table 4.1 shows the relation between the service level and the safety factor  $z$ .

**Table 4.1:** Service level and safety factor (Source: Simchi-Levi et al. 2003)

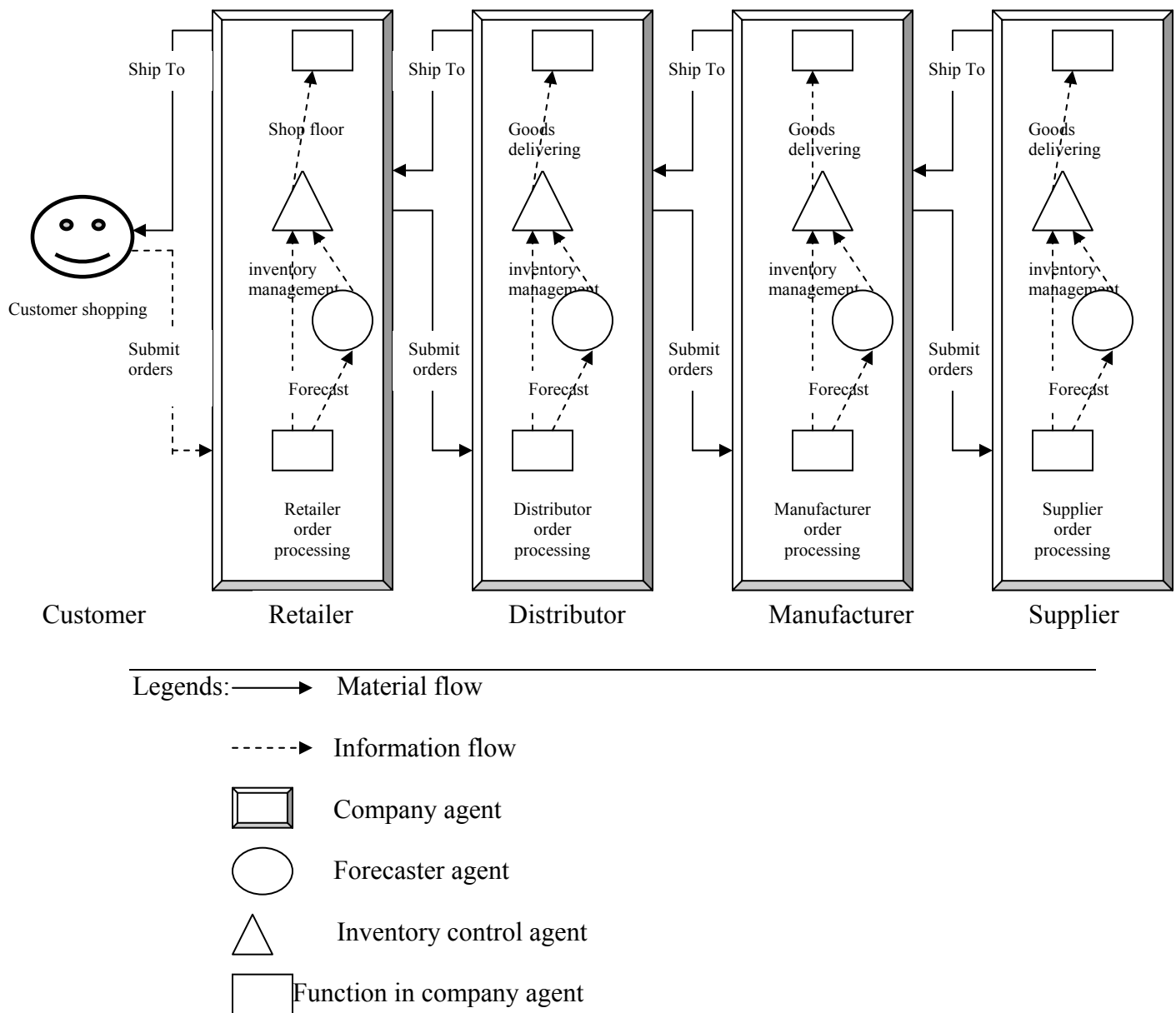
In each period, the inventory control agents ask the forecast agents for the estimated future customer demand and then calculate their target inventory levels in period  $t$  based on equation (4.2). After that, the inventory control agents check the inventory positions to determine the replenishment order size, and inform the company agents they belong to. The inventory position is the inventory-on-hand plus the products on-order minus backorders quantity:

$$\text{Inventory position} = \text{on hand (inventory level)} + \text{on order} - \text{backorders} \quad (4.4)$$

The replenishment order size is the target inventory level  $y_t$  minus the inventory position. It is given in formula (4.5).

$$\text{Replenishment order size} = \text{the target inventory level} - \text{inventory position} \quad (4.5)$$

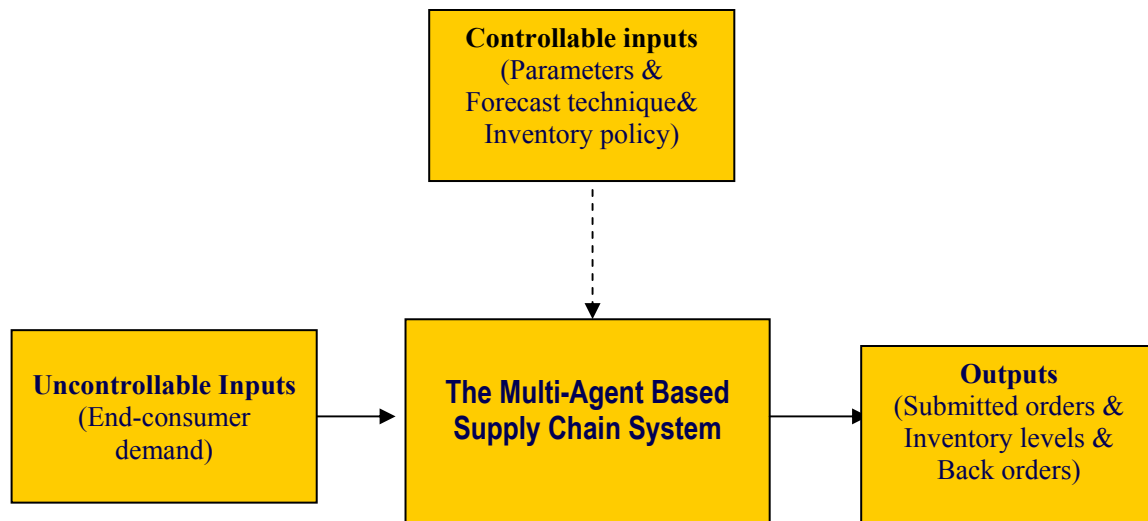
Figure 4.3 shows the three types of agents, as well as the information and material flows between the agents in the supply chain described above:



**Figure 4.3:** Supply chain prototype structure and processes

Besides the agents described above, there are two other agents in this prototype – the end-consumer agent and the simulation agent. The end-consumer agent generates the

demands, which are inputs of the multi-agent system. The generated end-consumer demands are the random variables that follow normal distribution functions with a mean and a variance (Law and Kelton 2000). The mean and the variance are the system parameters and are the controllable inputs of the system. The simulation agent represents functionality to run and analyze the supply chain. During the simulation, each company agent records its own relevant data every week to build a database, which will be communicated to a simulation agent at the end of the simulation for analysis.



**Figure 4.4:** Supply Chain Simulation Prototype

### 4.3 Performance measures and simulation scenarios

To study the impacts of the information sharing on supply chain performance, the following four scenarios are considered:

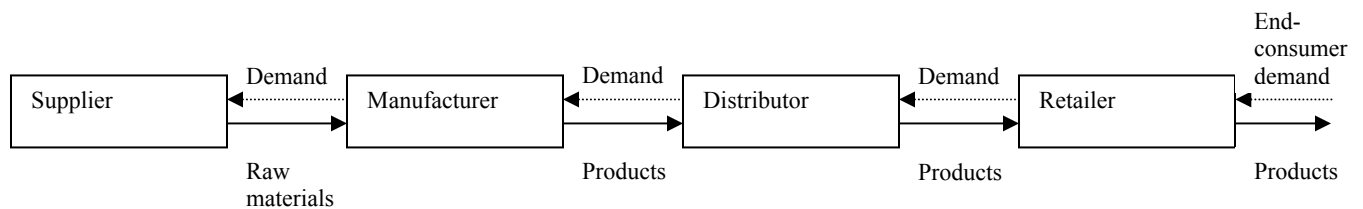
Scenario 1: No information sharing of end-consumer demand with minor customer demand fluctuations.

Scenario 2: No information sharing of end-consumer demand with major customer demand fluctuations.

Scenario 3: End-customer demand information sharing with minor customer demand fluctuations.

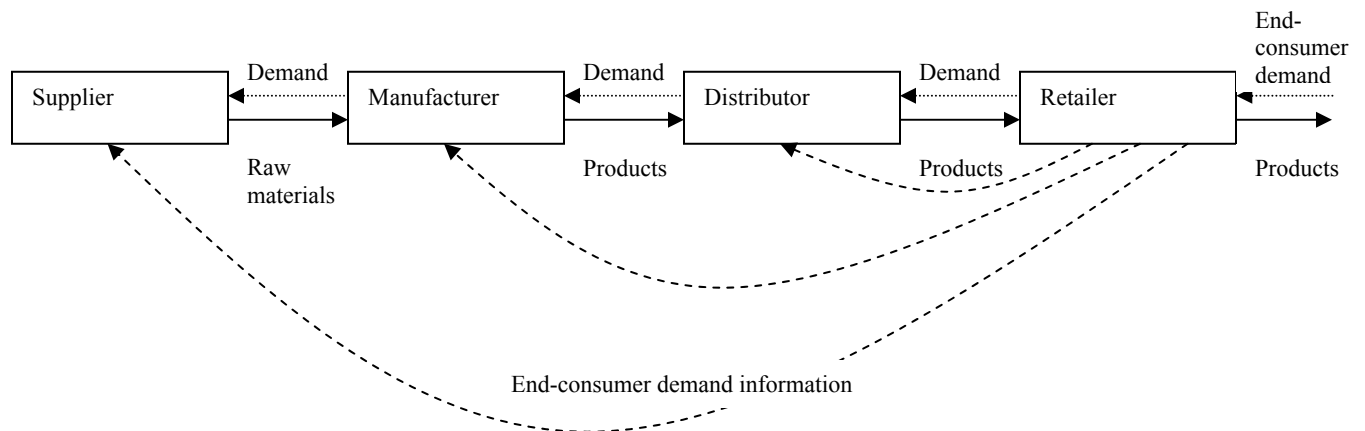
Scenario 4: End-customer demand information sharing with major customer demand fluctuations.

Figure 4.5 illustrates scenario 1 and 2 in which, the upstream echelon has no additional information from the down stream echelon except the current submitted order and the past echelon demand data. The upstream echelon uses past echelon demand data together with the current order from the downstream echelon to forecast the demand for future periods using a forecasting technique such as exponential smoothing.



**Figure 4.5:** Supply chain model in scenario 1 and 2.

Figure 4.6 illustrates scenarios 3 and 4, in which all echelons have access to past and current information on end-customer demand. Rather than using the historical data of the downstream echelon demand, the demand for the downstream echelon is forecasted using the historical end-customer demand data.

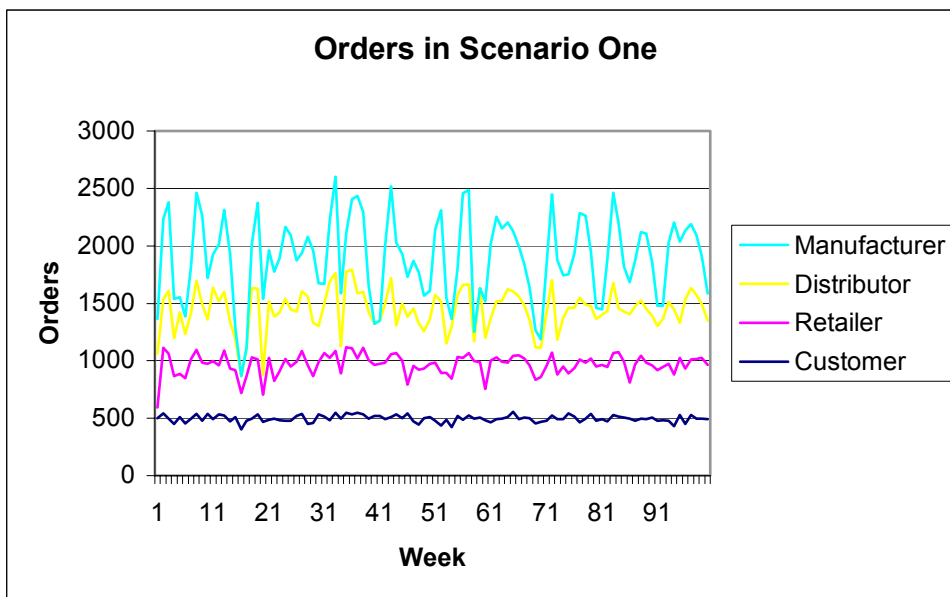


**Figure 4.6:** Supply chain model in scenarios 3 and 4.

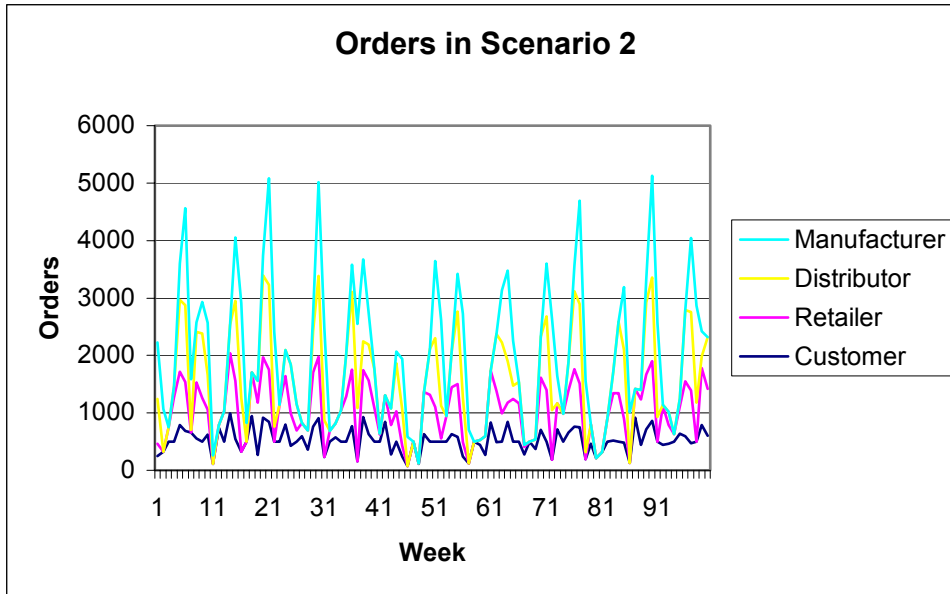
In this experiment, the end-consumer's demand is represented by the Normal distribution function. In scenarios 1 and 3, the mean of the end-consumer demand is assumed to be 500 and the standard deviation is 30. In scenarios 2 and 4, the end-consumer demand has the same mean and a deviation of 100 to indicate customer demand fluctuation. The order service level for each stage is assumed to be 99%. An exponential smoothing parameter of 0.3 is used. The simulation is run with discrete time intervals, the granularity is one week. We run the simulation for 250 weeks with the first 150 weeks to initialise the system. Data from the last 100 weeks is used to analyse the performance. We have focused on measuring the inventory cost and backorder penalties; i.e. the stock-out cost. Inventory cost and stock-out cost can be respectively expressed as a function of inventory level and backorder quantities. As a result, instead of measuring the inventory costs and backorder penalties directly, this experiment measures the inventory level and the backorder quantity. We also measure the order fluctuation.

## 4.4 Simulation Results

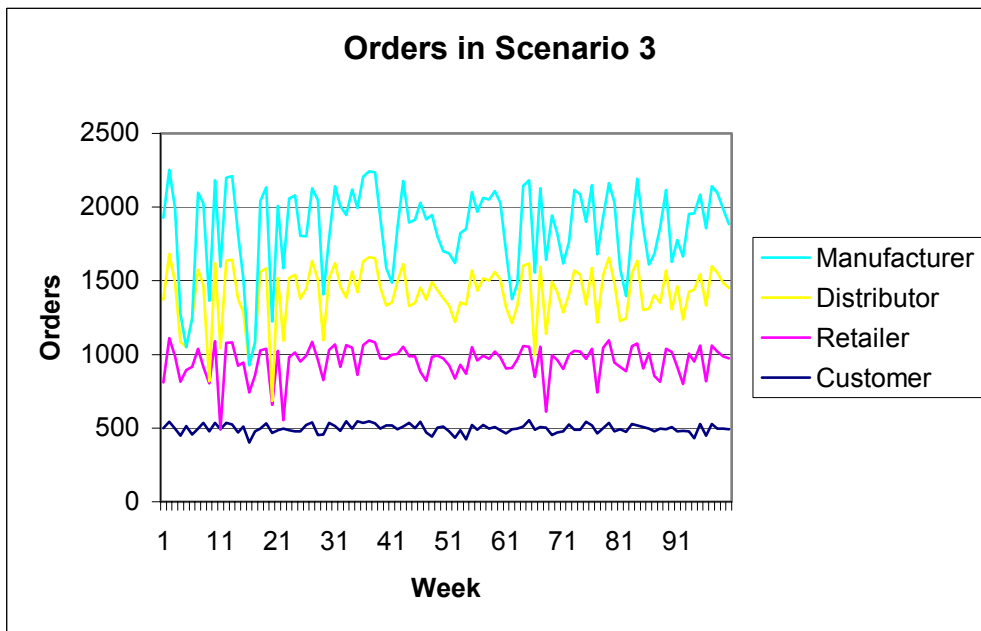
The results of the four simulation scenarios are presented in figures 4.7 to 4.18. Figures 4.7 to 4.10 present orders in the supply chain in four scenarios. In these figures, the lines in the graph respectively represent the end-consumer demand, retailer's orders, distributor's orders, and manufacturer's orders.



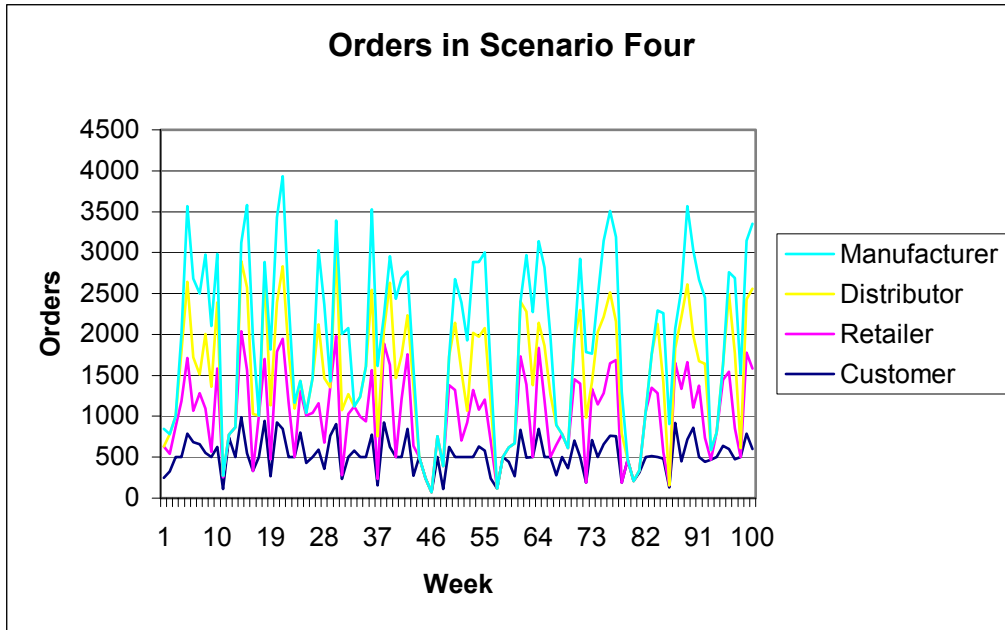
**Figure 4.7:** Orders in Scenario 1.



**Figure 4.8:** Orders in Scenario 2.

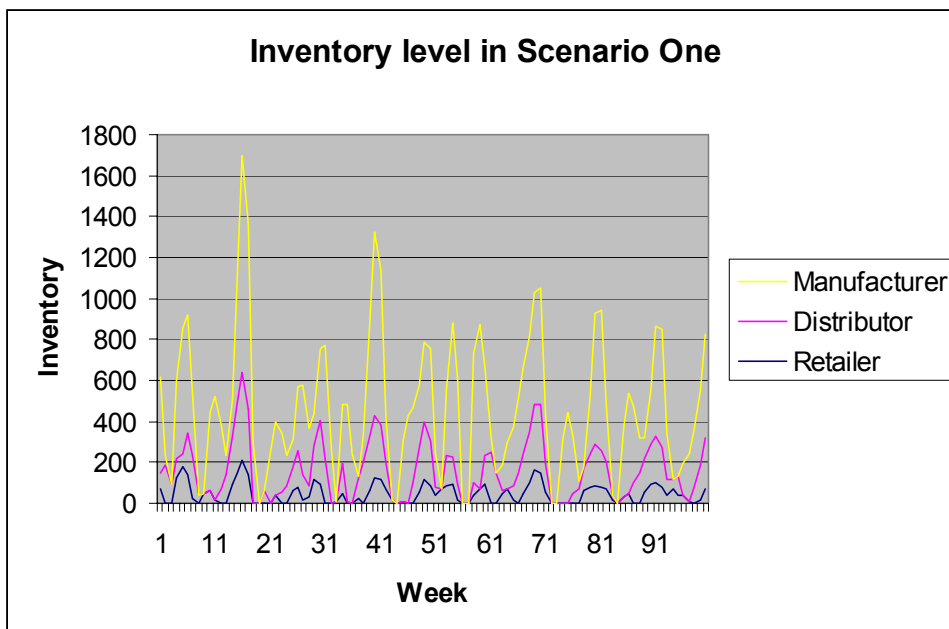


**Figure 4.9:** Orders in Scenario 3.

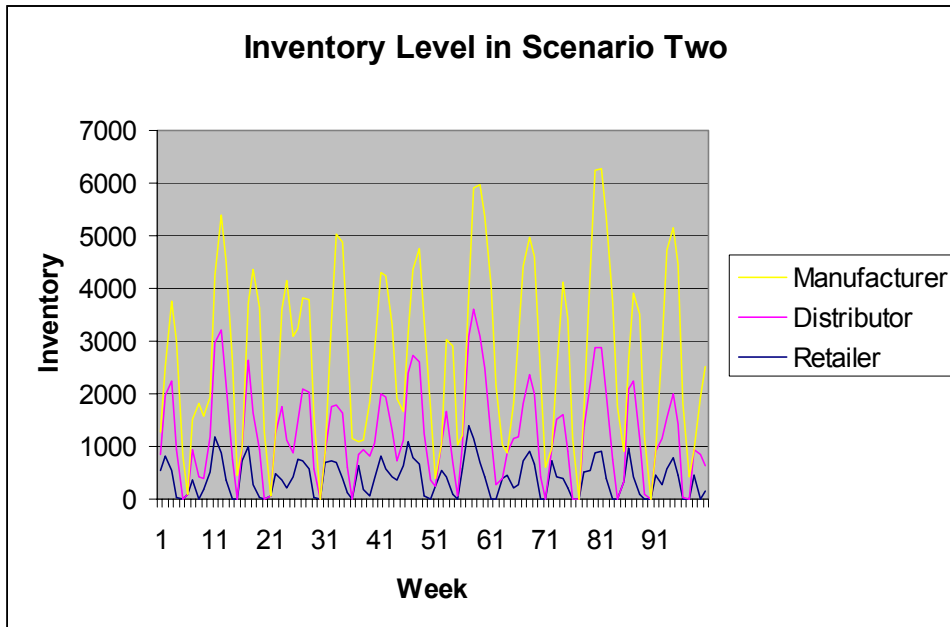


**Figure 4.10:** Orders in Scenario 4.

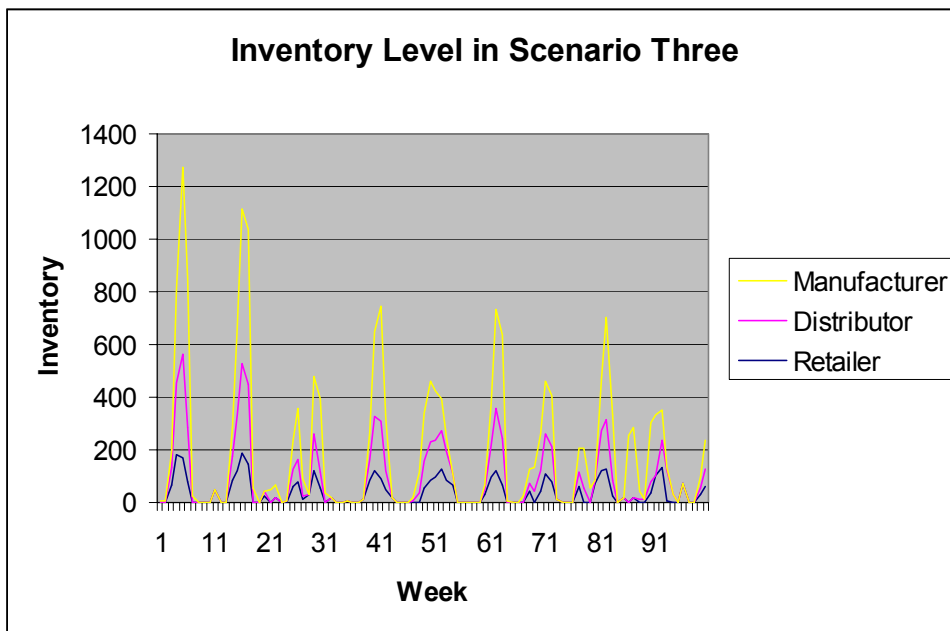
Figures 4.11 to 4.14 present the inventory levels in the supply chain in each of the four scenarios. In these figures, the lines in the graph respectively represent retailer's inventory level, distributor's inventory, and the manufacturer's inventory.



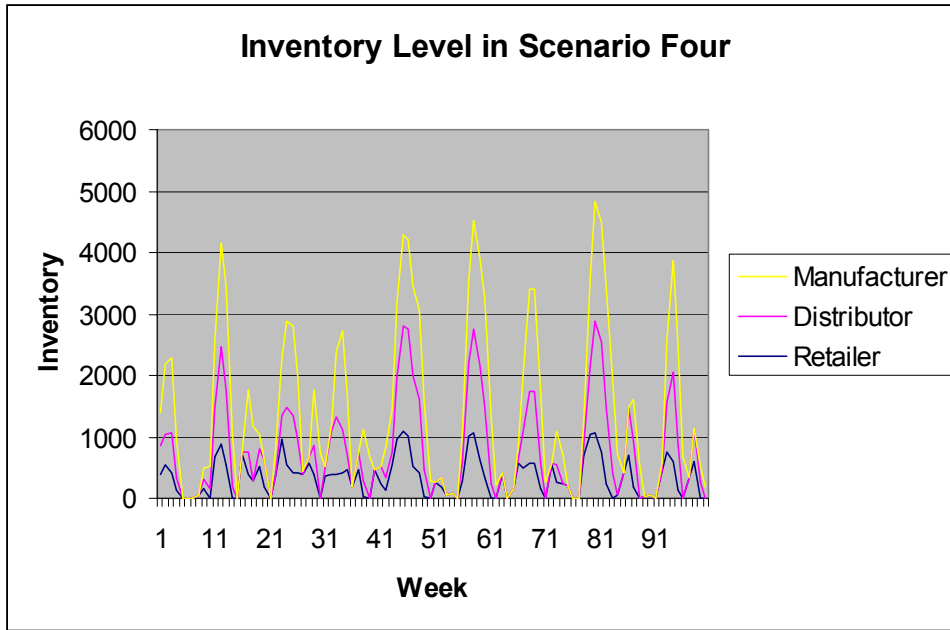
**Figure 4.11:** Inventory level in Scenario 1.



**Figure 4.12:** Inventory level in Scenario 2.

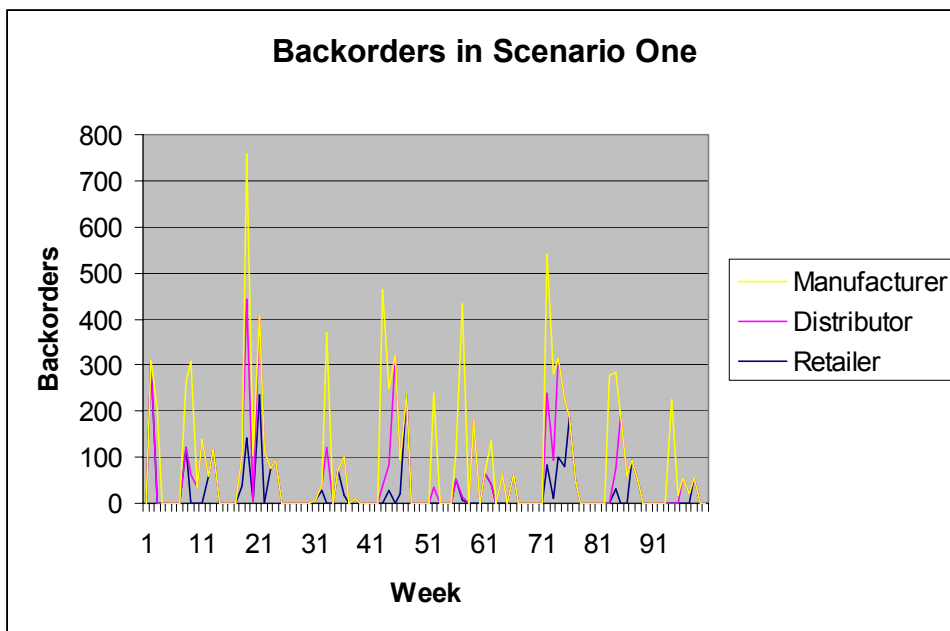


**Figure 4.13:** Inventory level in Scenario 3.

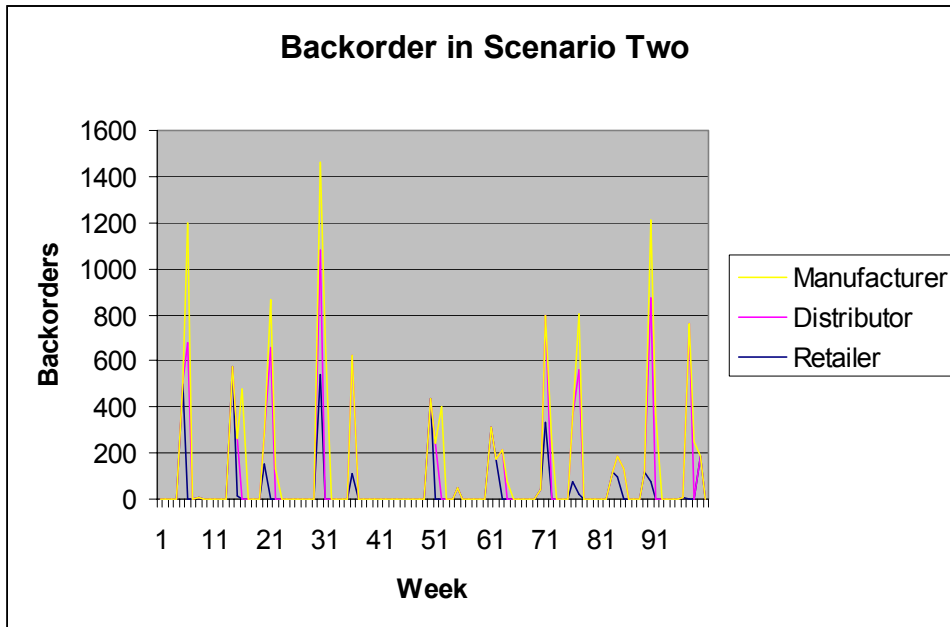


**Figure 4.14:** Inventory level in Scenario 4.

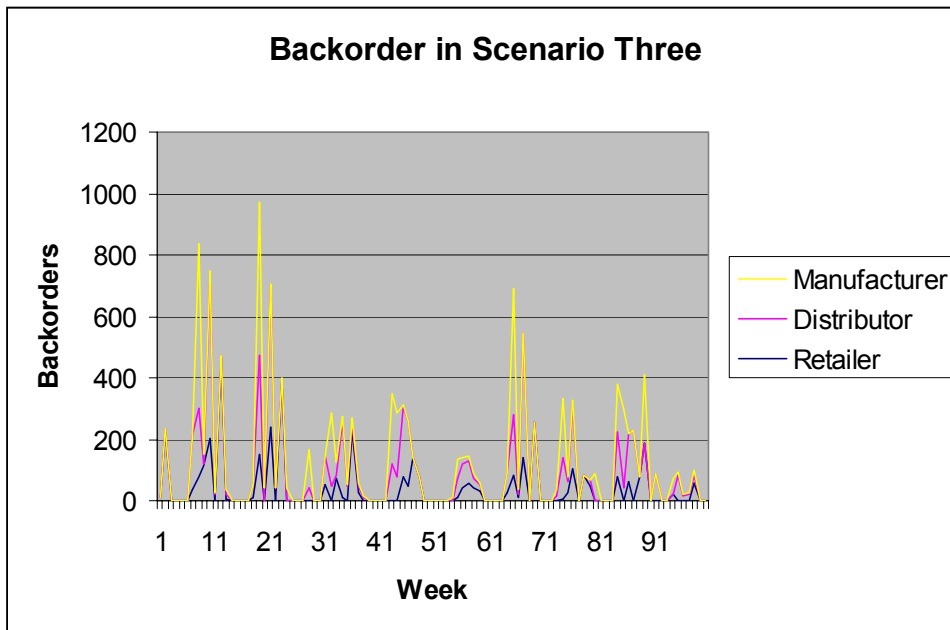
Similarly, figures 4.15 to 4.18 present the backorders in the supply chain. In these figures, the lines in the graph respectively represent retailer's stockout quantities, distributor's backorders, and manufacturer's backorders.



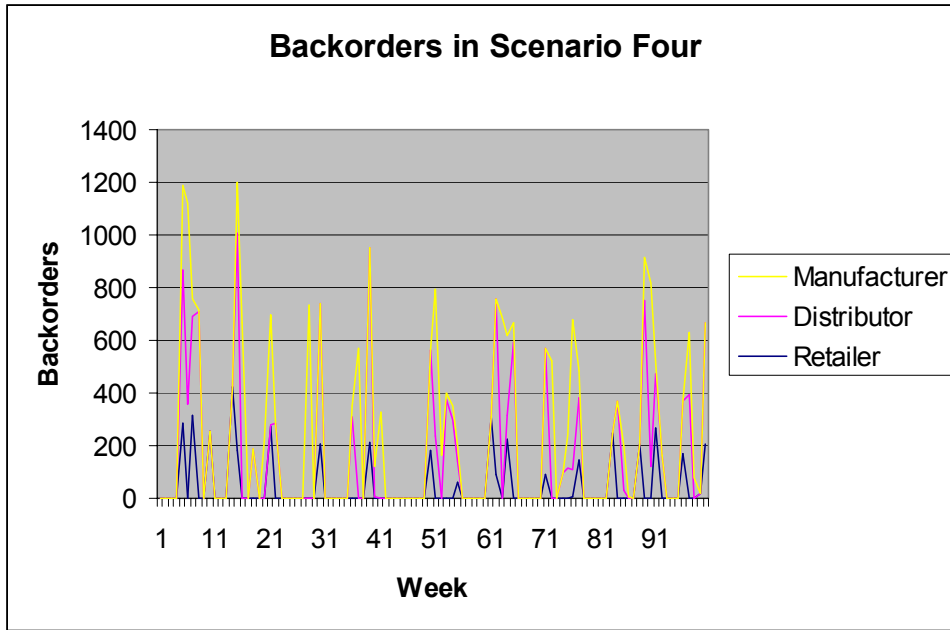
**Figure 4.15:** Backorders in Scenario 1.



**Figure 4.16:** Backorders in Scenario 2.



**Figure 4.17:** Backorders in Scenario 3.



**Figure 4.18:** Backorders in Scenario 4.

## 4.5 Discussion

Figures 4.7 to 4.10 present the variability of the orders placed by stages 1, 2, and 3 of the supply chain. From the results, we can see that as orders flow within the supply chain from the customer through the retailer, the distributor, the manufacturer, and then to the supplier, the orders' variability increases. This phenomenon of fluctuation of the order stream is known as the "bullwhip effect". As discussed in chapter two, the bullwhip effect is a major concern for many manufacturers, distributors and retailers because the increased variability in the order process requires each facility to increase its safety stocking throughout the systems. This results in increased costs due to overstocking throughout the systems, and can lead to an inefficient use of resources, such as labour and transportation. This effect is due to the fact that it is not clear whether resources should be planned based on the average order received by the facility or on maximum order.

Through the simulation, the bullwhip effect in the supply chain has been demonstrated clearly in figures 4.7 to 4.10. Comparing figures 4.7 and 4.8, as well as 4.9 and 4.10, we can see that the variability of the orders in the supply chain increases when the fluctuation of end-consumer demand increases.

We are interested in determining the impact of information sharing on supply chain performance, and more specifically on the bullwhip effect. To do this, we compare the variability at each stage of the supply chain for the scenarios with and without information sharing. We have already seen (in figures 4.7 to 4.10) that for all scenarios with information sharing and without information sharing, the variance of the order quantities becomes larger as we move up the supply chain. Similarly, from figures 4.11 to 4.14, we can see that in all scenarios, the variance of the inventory level and the variance of the backorder quantity become larger as we move up the supply chain from retailer to manufacturer. The difference in the supply chain with and without information sharing in terms of how much the variability grows as we move from stage to stage in the supply chain increases. The results shown in figures 4.7 to 4.10, and 4.11 to 4.14 indicate that, for supply chain with information sharing and customer demand information available at each stage of the supply chain, the increase in variability of orders and inventory levels at each stage is lower than a supply chain without information sharing, in which only the retailer knows the customer demand. This can be explained as follows. When end-consumer demand information is shared in the supply chain, each supply chain participant can use the actual customer demand data to estimate the future demand and the safety stock level. When demand information is not shared, each stage must use the

orders placed by the previous state to estimate future demand and decide on the safety stock level. As we have already seen, these orders are more varied than the actual customer demand data, and thus the forecasts created using these orders can vary, leading to even more variable orders and higher safety stock level. Therefore the results demonstrate that information sharing can reduce the bullwhip effect and the inventory level. However, as shown in these figures, information sharing will not eliminate the bullwhip effect. Furthermore, comparing figures 4.11 and 4.13 as well as 4.12 and 4.14, we can see that the increase in variability of backorder quantities for the supply chain with information sharing is higher than the supply chain without information sharing. This happens because when each stage in the supply chain uses end-consumer demand information instead of using the submitted order from the downstream entity to forecast the future demand, the variability of the order and the safety stock level decreases. As a result, the backorder quantities are increased.

Therefore, although information sharing as a supply chain collaboration strategy can reduce the bullwhip effect and the inventory level significantly, however it can also increase the backorder quantities, which in turn can increase the stock-out costs. In general, the total cost includes the ordering cost, inventory holding cost, and the stock-out cost. The ordering cost is the expense of issuing a purchase order to the outside supplier; the inventory holding cost subsumes the costs associated with investing in inventory and maintaining the physical investment in storage; the stock-out cost is the economic consequence of an external or an internal shortage. The goal of supply chain management is to minimize the total cost of the supply chain. Therefore, break-even analysis between

the stock out costs and the inventory holding costs need to be considered. Thus, only sharing information may not be enough to achieve the best supply chain performance, more efforts are required to study on how to use the information shared to improve supply chain forecasting and planning.

## **4.6 Summary**

In this chapter, we have discussed the development of a simulation prototype for modelling and analysing a supply chain. The multi-agent modelling techniques are applied to describe the supply chain components and behaviors. The model consists of four communication agents and six function agents. Four communication agents named as the company agents were developed to describe the supply chain entities. They are designed to facilitate the interaction among the supply chain partners and conduct the functions of ordering and products delivery. Six function agents have been developed to conduct the forecasting and inventory management functions. The developed prototype facilitates the study of the value of information sharing on the supply chain performance. Four scenarios are designed for this research and the performance measure focuses on both the inventory cost and the customer service levels. The simulation results show that information sharing as a basic supply chain collaboration strategy can reduce the bullwhip effect and results in lower amounts of the inventory holding but it also leads to higher stock outs.

## **Chapter V Conclusion**

This chapter concludes the presentation of the thesis. The chapter is organised as follows.

Section 1 summaries the research and outlines research contribution, followed by future directions in Section 2.

### **5.1 Research summary**

This research aims to investigate the application of a multi-agent approach to model and simulate the supply chain with special focus on information sharing as a basic supply chain collaboration strategy. The literature in the areas of supply chains, supply chain information systems and multi-agent systems has been reviewed. A supply chain is defined as a network of suppliers, factories, warehouses, distribution centers and retailers through which raw materials are acquired, transformed into products which are then delivered to customers. This type of supply chain network in general involves heterogeneous environments. To optimize supply chain performance at the system level, the SCM demands process integration within a company and the collaboration of planning and actions among other supply chain partners. Various supply chain information systems have been developed to facilitate the SCM and to enable process integration. However, current information systems only support collaboration between two levels of a supply chain and not on the system-wide level. System-wide coordination and collaboration, which is not covered sufficiently by traditional information systems, requires more effective modeling methods and information technology support. Intelligent agent technology and multi-agent systems have shown great potential in

overcoming many limitations of current supply chain technologies. There are many similar characteristics between multi-agent systems and supply chain systems, such as complexity, modularity, distribution, and reconfigurativity. It is argued that the supply chain is a well-suited application domain for multi-agent systems.

We have developed a multi-agent based collaborative supply chain management system conceptual framework. This framework describes the adoption of a multi-agent approach to achieve supply chain collaboration and to support information sharing, collaborative decision-making and coordinated problem solving. We believe the proposed multi-agent based supply chain system, which combines the capabilities of current available information technology and information systems, is able to support supply chain management.

A simulation model has been developed to model a supply chain. In the prototype, four communication agents and six function agents were designed to describe the supply chain components and behaviors. The prototype was used as a simulation tool to analyse the impact of information sharing, which is a basic supply chain collaboration strategy, on supply chain performance. We have compared the effects of information sharing and demand variation on the performance of a four-stage supply chain multi-agent model. The performance measures that were used include order variation, inventory level and backorder quantity. Based on the results of our experiments, we conclude that information sharing as a basic supply chain collaboration strategy can reduce the

bullwhip effect and results in lower amounts of the inventory holding but it also leads to higher stock outs.

## **5.2 Future research direction**

For future work, we suggest to extend the multi-agent model from the simple four-echelon supply chain structure to larger supply chain networks with more entities in each echelon and to use the model to experiment with different assumptions and alternative measures of performance. For example, when analyzing information sharing in this research, lead times were assumed to be constant. The effect of stochastic lead times should be studied in future research. Furthermore, there is a need for analyzing the performance of the prototype with various demand patterns in markets. This research has focused on analysing end-consumer demand information sharing as the information sharing strategy. It will be interesting to evaluate the effect of different information sharing strategies, such as real-time inventory level information sharing, on the performance of the supply chain and to determine how choices of operating and communication policies affect variability from retailer to manufacturer and the total cost of the systems.

## References

- Baumgaertel, H., Brueckner, S., Parunak, V., Vanderbok, R., & Wilke, J. 2001, Agent models of supply networks dynamics Analysis, Design, and Operation. URL [http://www.anteaters.net/~sbrueckner/publications/2001/AgentModelsOfSupplyNetworkDynamics\\_submitted\\_20010209.pdf](http://www.anteaters.net/~sbrueckner/publications/2001/AgentModelsOfSupplyNetworkDynamics_submitted_20010209.pdf)
- Bowersox, D. J., Closs D. J. & Cooper, M. B. 2002, *Supply chain logistics management*, McGraw-Hill, Boston.
- Berger, A. J. 2003, 'e-commerce and supply chains-breaking down the boundaries', in Gattorna, J. L. (eds.), *Handbook of Supply Chain Management*, 5th edn, Gower, England, pp. 429-441
- Christopher, M. 1998, *Logistics and Supply Chain Management, Strategies for Reducing Cost and Improving Service*, 2<sup>nd</sup> edn, Prentice Hall, Great Britain.
- Chopra, S. & Meindl, P. 2004, *Supply chain Management: Strategy, Planning and Operation*, 2nd edn, Upper Saddle River, New Jersey.
- Emiliani, M. L. 2000, 'Business-To-Business Online Auctions: Key Issues for Purchasing process Improvement', *Supply Chain Management*, vol.5, no. 4, pp.176-186.
- Erasala, N., Yen, D. C. & Rajkumar, T. M. 2003, 'Enterprise Application Integration in the electronic commerce world', *Computer standards & Interfaces*, vol. 25, pp.69-82.
- Finin, T. & Labrou, Y. 1997, 'KWML as an agent communication language', In: *Software Agents*, Bradshaw, J.M. (ed.), MIT Press Cambridge, London, pp.205-223.
- Frey, D., Stockheim, T., Woelk, P. & Zimmermann, P. 2003, 'Integrated multi-agent-based supply chain management', *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*.
- Genesereth, M.R. & Ketchpel, S. P. 1994, 'Software agents', *Communications of the ACM*, vol. 37, no.7, pp.48-53.
- Gattorna, J. L. 2000, 'The E-Supply Chain Reaches Asian Shores', *ASCET*, vol. 2, pp. 335-339.
- Gupta, A., Whitman, L. & Agarwal, R. K. 2001, 'Supply chain agent decision aid system (SCADAS)', *Proceedings of the 2001 Winter Simulation Conference*, pp.553-559.
- Gattorna, J.L., Ogulin R. & Reynolds, M. K. (eds.) 2003, *Handbook of supply chain management*, 5<sup>th</sup> edn, Gower, USA.

Hax, A.C. & Candea, D. 1984, *Production and inventory management*, Prentice-Hall Inc, Englewood Cliffs.

Huhns, M. N. & Stephens, L. M. 1999, 'Multi-agent systems and societies of agents', In *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, Weiss, G. (ed.), pp. 79–120. MIT Press, Cambridge.

Holmström, F., Framling, K., Kaipia, R. & Saranen, F. 2002, 'Collaborative planning forecasting and replenishment: new solutions needed for mass collaboration', *Supply Chain Management: An International Journal*, vol. 7, no. 3, pp.136-145.

Iskanius, P., Helaakoski, H., Alarukka, A. M. & Kipina, J. 2004, 'Transparent Information Flow in Business Networks by using Agents', *IEEE International Engineering Management Conference 2004*, pp.1342-1346.

Jennings, N. R., Sycara, K. & Wooldridge, M. 1998, 'A Roadmap of Agent Research and Development', *Autonomous Agents and Multi-Agent Systems*, vol. 1, no. 1, pp. 7-38.

Jennings, N. 2000, 'On agent-based software engineering', *Artificial Intelligence*, vol. 117, no. 2, pp. 277-296.

Jennings, N. R. & Wooldridge, M. J. 2002, 'Applications of Intelligent Agents, In Jennings, N. R. & Wooldridge, M. J. (eds.) *Agent Technology Foundations, Applications, and Markets*, Springer, New York.

Kumar, P. & Kumar, M., 2003, Vendor Managed Inventory in Retail Industry, Tata Consultancy Services white paper, URL [http://www.tatainfotech.com/0\\_industry\\_practices/retail/download/VMI\\_white\\_paper.pdf](http://www.tatainfotech.com/0_industry_practices/retail/download/VMI_white_paper.pdf) Accessed 2 November 2005.

Lee, H. L., Padmanadhan, V. & Whang, S. 1997, 'The Bullwhip Effect in Supply Chains', *Operations Management and Research*, vol. 38, no. 3, pp. 93-102.

Lee, K, Kim, W. & Kim, M. 2004, 'Supply Chain Management Using Multi-agent System', In *CIA 2004, LNAI 3191*, Klusch, M. (eds.), Springer-Verlag, Berlin Heidelberg, pp. 215-225.

Law, A. M. & Kelton, W. D. 2000, *Simulation Modeling and Analysis*, 3rd edn, McGraw-Hill, New York.

Mukhopadhyay, T., Kekre, S. & Kalathur, S. 1995, "Business value of information technology: a study of electronic data interchange", *MIS Quarterly*, vol. 19, no. 2, pp.137-156.

Mentzer, J. T., Foggin, T. H. & Golicic, S. 2000, 'Collaboration: The Enablers, Impediments, and Benefits', *Supply Chain Management Review*, Sep/Oct, pp. 52-58.

McCullen, P. & Towill, D. 2002, 'Diagnosis and reduction of bullwhip in supply chains', *Supply Chain Management: An International Journal*, vol. 7, no 3, pp. 164-179.

Matchette, J. & Seikel, A. 2004, 'How to win friends and influence supply chain leaders', URL <http://www.logisticstoday.com/displayStory.asp?sNO=6828&pNum=1&OASKEY=CurrentIssue>, Accessed 31 March 2005.

Marik, V. & McFarlane, D. 2005, 'Industrial Adoption of Agent-Based Technologies', *IEEE Intelligent System*, vol. 20, no. 1, pp.27- 35

Nwana, H. S. 1996, 'Software Agents: An Overview', *Knowledge Engineering Review*, vol. 11, no. 3, pp. 205-244.

Neef, D. 2001, *e-Procurement from Strategy to Implementation*, Prentice Hall PTR, USA.

Parunak, H.V.D., Savit, R. & Riolo, R. L. 1998, 'Agent-based Modeling vs. Equation-Based Modeling: A Vase Study and Users' Guide', in *Multi-Agent Systems and Agent-based Simulation* Sichman, J.S., Conte, R. & Gilbert, N. (eds.), Springer, pp.10-26.

Parunak, H. V. D., 1998, 'Practical and Industrial Applications of Agent-Based Systems', URL <http://agents.umbc.edu/papers/apps98.pdf>, Accessed 31 March 2005.

Parunak, H. V. D. & VanderBok, R. 1998, 'Modeling the extended supply network', in *ISA-Tech'98 (Houston)*, Industrial Technology Institute.

Parunak, H. V. D., 2000, 'A Practitioners' Review of Industrial Agent Applications', *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 4, pp. 389-407.

Puckridge, D. S. & Woolsey, I. 2003 'Information systems strategy for supply chains', in *Handbook of Supply Chain Management* Gattorna, J. L. (eds.), 5th edition, Gower, England, pp. 406-425.

Quinn, F. J. 2001, 'Collaboration: More than just technology', *ASCET*, vol. 3.

Riggin, F. J. & Mulhopadhaya 1994, 'Interdependent benefits from interorganizational systems: Opportunities from business partner reengineering', *Journal of Management and Information Systems*, vol.11, no. 2, pp.37-57.

Raghunathan, S. & Yeh, A. B. 2001, 'Beyond EDI: Impact of Continuous Replenishment Program (CRP) Between a Manufacturer and its Retailers', *Information Systems Research*, vol. 12, no. 4, pp.406-419.

Shoham, Y. 1993, 'Agent –oriented programming', *Artificial Intelligence*, vol. 60, no. 1, pp. 51-92.

Srinivasan, K., Kekre, S. & Mukhopadhyay, T. 1994, 'Impact of electronic data interchange technology on JIT shipments', *Management Science*, vol. 40, no. 10, pp. 1281-1304.

Sycara, K. P., 1998, 'Multiagent Systems', *AI Magazine*, vol. 19, no. 2, pp. 79-92.

Swaminathan, J. M., Smith, S. F & Sadeh, N. M. 1998, 'Modeling Supply Chain Dynamics: A Multiagent Approach', *Decision Science*, vol. 29, no. 3, pp.607-632.

Sadeh, N.M., Hildum, D. W., Kjenstad, D. & Tseng, A. 2001, 'MASCOT: an agent-based architecture for dynamic supply chain creation and coordination in the internet economy', *Production Planning & Control*, vol. 12, no. 3, pp.212-223.

Shapiro, J. F. 2001, *Modeling the supply chain*, Brooks/Cole-Thomson Learning, Pacific Grove.

Shapiro, J. F. 2003, 'Bottom-Ups vs. Top-Down Approaches to Supply Chain Modeling', in *Quantitative models for supply chain management*, Tayur, S. & Ganeshan, R. & Magazine, M. (eds.), Kluwer Academic Publisher, London.

Simchi-Levi, D., Kaminsky, P. & Simchi-Levi, E. 2003, *Designing and Managing the Supply Chain: concepts, strategies and case studies*, 2<sup>nd</sup> edn, McGraw-Hill, New York.

Simchi-Levi, D & Simchi-Levi, E. 2001, 'The Dramatic Impact of Internet on Supply Chain', *ASCET*, vol. 3, 174-182.

Stadtler, H. & Kilger, C., (Eds) 2002, 'Supply Chain Management and Advanced Planning: Concepts, Models', *Software and Case Studies*, 2<sup>rd</sup> edn, Springer, New York.

Supply Chain Council 2001, SCOR Version 5.0! Introduction web cast. [http://www.supplychainworld.org/WebCast/SCOR50\\_overview.ppt](http://www.supplychainworld.org/WebCast/SCOR50_overview.ppt).

Sanchez, A. M. & Perez, M. P. 2003, 'The use of EDI for interorganisational cooperation and coordination in the supply chain', *Integrated Manufacturing Systems*, vol. 14, no. 8, pp. 642-651.

Swaminathan, J.M. & Tayur, S.R 2003, 'Models for supply chains in E-business', *Management Science*, vol. 49, no.10, pp.1387-1406.

Tersine, R. J. 1994, *Principles of Inventory and Materials Management*, 4<sup>th</sup> edn, Prentice-Hall, New Jersey.

Taylor, D. A. 2004, *Supply chains: a manager's guide*, Addison-Wesley, Boston.

Umble, E. J., Haft, R. R. & Umble, M.M. 2003, 'Enterprise resource planning: Implementation procedures and critical success factors', *European Journal of Operational Research*, vol. 146, pp. 241-257.

Verdicchio, M. & Colombetti, M. 2002, 'Commitments for Agent-Based Supply Chain Management', *ACM SIGecom Exchanges*, vol. 3, no. 1, pp. 13-23.

VICS Association 2006, Web site. <http://www.vics.org/committees/cpfr/> Accessed 2<sup>nd</sup> February, 2006.

Wang, E.T.G. & Seidmann, A. 1995, 'Electronic data interchange: Competitive externalities and strategic implementation policies', *Management Science*, vol. 31, no. 3, pp.401-418.

Wooldridge, M. & Jennings, N., 1995, 'Intelligent Agents: Theory and Practice', *Knowledge Engineering Review*, vol. 10, no. 2, pp.115-152.

Walton, S. V. & Gupta, J. N. D., 1999, 'Electronic data interchange for process change in an integrated supply chain', *International Journal of Operations & Production Management*, vol. 19, no. 4, pp. 372-388.

Wooldridge, M., 2002, *An Introduction to Multi-Agent Systems*, John Wiley & Sons, Inc. New York.

Wooldridge, M. & Ciancarini, 'Agent-Oriented Software Engineering: The State of the Art', URL <http://www.csc.liv.ac.uk/~mjw/pubs/aose2000a.pdf>, Accessed 31 March, 2005.

Xue, X., Li, X., Shen, Q. & Wang, Y. 2005, 'An agent-based framework for supply chain coordination in construction', *Automation in Construction*, vol. 14, pp. 413-430.

Yu, Z., Yan, H. & Cheng, T.C.E. 2001, 'Benefits of information sharing with supply chain partnerships', *Industrial Management & Data systems*, vol. 103, no. 3, pp. 114-119

Yuan, Y., Liang, T. P. & Zhang, J.J 2001, 'Using Agent technology to support supply chain management: Potentials and challenges', McMaster University, Michael G. DeGroote School of Business, *Working Paper* Series no 453.

## **Appendix A - Program Codes**

The order of files is in alphabetical ascending.

Customer.java

Distributor.java

DistributorForecaster.java

Liner.java

Logger.java

Main.java

Manufacturer.java

ManufacturerForecaster.java

Retailer.java

RetailerForecaster.java

Supplier.java

Transaction.java

## Customer.java

```
//import java.io.File;
//import java.io.FileInputStream;
//import java.io.FileNotFoundException;
//import java.io.IOException;
//import java.util.Properties;
import java.util.Vector;
import java.util.Random;

public class Customer {

    int [] orderByWeek; // record the end user orders, which are generated
                        // by the Class-CustOrderGenerator to the
array

    Random rand = new Random();
    Vector orders;
    Retailer retailer;
    public Customer(int [] corders) {
        orderByWeek = corders;
        orders = new Vector();

        retailer = null;
    }

    public void setRetailer(Retailer r) { retailer = r; }

    public void submitOrder(int w) {
        int qty = orderByWeek[w];
        // create new order
        Transaction order = new Transaction(w, qty);
        // submit order
        retailer.takeOrder(order);
        // keep track of what have been ordered
        orders.add(order);
        System.out.println("\nCustomer Week:"+w+",
submittedOrder:"+order.toString());
    }
}
```

## Distributer.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;
import java.util.Vector;

public class Distributor {

    /** Creates a new instance of Distributor */
    Vector retlOrders; //record weekly real retailer demand
    //Vector relOrders_for_Forecasting; //record real retailer demand only when the
demand is not zero (dt real retailer demand in week t.)

    Vector dtStar; //record forecasted weekly retailer demand
    //Vector dtStar_for_Forecasting; // record forecasted retailer demand only when
the data is not zero.

    Vector orders; //Replenishment orders from DC to MF
    Vector stockLevels; //Current (updated) Inventory level
    //Vector stockOuts;
    Vector backOrders_R_DC; //record weekly backorder quantity between Retailer and
DC.
    Vector weeklyCosts;

    double dcOrders_MEAN; //the MEAN of replenishment orders from retailer to
DC
    double dcOrders_MAD; //the MAD of replenishment orders from retailer to DC
    double dcInventorylevel_MEAN;
    double dcInventorylevel_MAD;
    double dcBackorder_MEAN;
    double dcBackorder_MAD;
    double dcWeeklycost_MEAN;
    double dcWeeklycost_MAD;
```

```
int backOrderQty_R_DC; //updated backorder quantity between Retailer and DC.
int backOrderQty_DC_MF; //updated backorder quantity between DC and MF.
```

```
Transaction weeklyCost;
    Transaction submittedOrder;
Retailer retailer;
Manufacturer manufacturer;
    int qtyReceivedFromMF;
    Transaction stockLevel;
    int[] bufferFromMF; //the size of buffer is leadtimeofMF+1
    String propFile;
    DistributorForecaster forecaster;
```

```
double ORDER_COST;
double UNIT_PURCHASE_COST;
double HOLDING_COST;
int LEAD_TIME_FROMMF;
double STOCKOUT_COST_PERUNIT;
int INITIAL_INVENTORY_LEVEL;
double SAFETY_STOCK_PARAMETER;
double ALPHA;
```

```
public Distributor (String pf) {
    Properties prop = new Properties();
    try {
        prop.load(new FileInputStream(new File(pf)));

        ORDER_COST =
Double.parseDouble(prop.getProperty("ORDER_COST"));
        UNIT_PURCHASE_COST =
Double.parseDouble(prop.getProperty("UNIT_PURCHASE_COST"));
        HOLDING_COST =
Double.parseDouble(prop.getProperty("HOLDING_COST"));
        LEAD_TIME_FROMMF =
Integer.parseInt(prop.getProperty("LEAD_TIME_FROMMF"));
        STOCKOUT_COST_PERUNIT =
Double.parseDouble(prop.getProperty("STOCK_COST"));
        INITIAL_INVENTORY_LEVEL =
Integer.parseInt(prop.getProperty("INITIAL_INVENTORY_LEVEL"));
        SAFETY_STOCK_PARAMETER =
Double.parseDouble(prop.getProperty("SAFETY_STOCK_PARAMETER"));
        ALPHA = Double.parseDouble(prop.getProperty("ALPHA"));
```

```

        forecaster = new DistributorForecaster(ORDER_COST,
UNIT_PURCHASE_COST, HOLDING_COST, LEAD_TIME_FROMMF,
STOCKOUT_COST_PERUNIT, SAFETY_STOCK_PARAMETER, ALPHA);

```

```

    } catch(FileNotFoundException e){
        e.printStackTrace();
    } catch(IOException e){
        e.printStackTrace();
    } catch(NumberFormatException e){
        e.printStackTrace();
    }

```

```

retlOrders = new Vector();
dtStar = new Vector();
orders = new Vector();
stockLevels = new Vector();
backOrders_R_DC = new Vector();
weeklyCosts = new Vector();

```

```

submittedOrder = null;
retailer = null;
manufacturer = null;
qtyReceivedFromMF = 0;
stockLevel = null;
bufferFromMF = new int[LEAD_TIME_FROMMF+1];
backOrderQty_R_DC = 0;
propFile = pf;

```

```

dcOrders_MEAN = 0; //the MEAN of replenishment orders from retailer to DC
dcOrders_MAD = 0; //the MAD of replenishment orders from retailer to DC
dcInventorylevel_MEAN = 0;
dcInventorylevel_MAD = 0;
dcBackorder_MEAN = 0;
dcBackorder_MAD = 0;
dcWeeklycost_MEAN = 0;
dcWeeklycost_MAD = 0;
}

```

```

public void setRetailer(Retailer r) { retailer = r; }
public void setManufactuer(Manufacturer r) {manufacturer = r;}

```

```

public void takeOrder(Transaction o) {
submittedOrder = new Transaction(o.week, o.qty);

```

```

retlOrders.add(submittedOrder);

System.err.println("new retailer's order added: size= "+retlOrders.size());
}

// receive scheduled supply from MF and update stock level
public void receiveScheduledSupply(int w) {

    //take the front element from the buffer
    int qtyReceivedFromMF=bufferFromMF[0];
    //update the buffer push qty forward everyweel
    for(int i=0; i<bufferFromMF.length-1; i++){
        bufferFromMF[i] = bufferFromMF[i+1];
    }
    bufferFromMF[bufferFromMF.length-1] = 0;

    // initialize current level in stock
    int level = 0;
    // carry out the current stock level with last week's one
    if(w==1) {
        // Get the initial inventory level from the user
        level = INITIAL_INVENTORY_LEVEL;
    } else {
        level = ((Transaction)stockLevels.elementAt(w-2)).qty;
    }
    System.out.println("DC-Week:"+w+",initial Level from last
week:"+level+
        ", qty of supply from MF:"+qtyReceivedFromMF);
    // update the current stock level with qty received from MF
    stockLevel = new Transaction(w, level+qtyReceivedFromMF);
    System.out.println("DC-Week:"+w+", updated stockLevel BEFORE
processing order:"+stockLevel.toString());
}

// process order received from retailer
public void processOrders(int w) {

    if(submittedOrder.qty + backOrderQty_R_DC >=stockLevel.qty) {
        // update retailer's buffer
        retailer.insertDCSupplyIntoBuffer(stockLevel.qty);
        System.out.println("DC->=, submittedOrder.qty:"+submittedOrder.qty+
            ", stockLevel.qty: "+stockLevel.qty);
        backOrderQty_R_DC = submittedOrder.qty + backOrderQty_R_DC-
stockLevel.qty;
        stockLevel.qty = 0;
    }
}

```

```

    } else {
        // update retailer's buffer
        retailer.insertDCSupplyIntoBuffer(submittedOrder.qty+backOrderQty_R_DC);
        System.out.println("DC-<, submittedOrder.qty:"+submittedOrder.qty+
            ", stockLevel.qty"+stockLevel.qty);
        stockLevel.qty -= (submittedOrder.qty + backOrderQty_R_DC);
        backOrderQty_R_DC = 0;
    }

    // inform retailer about the backorder of the order
    retailer.updateBackorder(backOrderQty_R_DC);
    System.out.println("DC backOrder to Retailer:"+backOrderQty_R_DC);

    // update distributor's stockLevels
    stockLevels.add(stockLevel);
    System.out.println("DC-Week:"+w+", updated stockLevel AFTER processing
order:"+stockLevel.toString());

    //record distributor's backorder level (between retailer & DC)
    Transaction backOrder = new Transaction(w, backOrderQty_R_DC);
    backOrders_R_DC.add(backOrder);

}

//submit order to MF (Every Tstar weeks)
public void submitOrder(int w, int inf) {

    /**setup dtstar (forecasted retailer demand in week t)
    *for week 1 to be actual retailer order of week 1
    *
    */

    /**no information sharing scenario
    * inf == 0: INFORMATIONSHARING ==0
    *
    */
    if(inf==0){
        System.err.println("No information sharing scenario!!");

        int dtstar1 = 0;

        if(w==1){
            dtStar.add(new Integer(

```

```

                ((Transaction)retlOrders.elementAt(w-1)).qty)
            );
            dtstar1 =
forecaster.computeDstar1(((Transaction)retlOrders.elementAt(w-1)).qty,
                ((Integer)dtStar.elementAt(w-1)).intValue());
        } else if(w > 1){
            dtstar1 = forecaster.computeDstar1(((Transaction)retlOrders.elementAt(w-
1)).qty,
                ((Integer)dtStar.elementAt(w-1)).intValue());
        }

        dtStar.add(new Integer(dtstar1));
        System.out.println("DC-week: " + w+", forecasted demand for next week-
no inforamtion sharing: "+dtstar1);

        /** Compute qty for this order
         * @param dtstar1: forecasted retailer demand in week t+1
         * @param et: maximum inventory target
         */

        double otstar=0.0;
        otstar = forecaster.computeOtstar(dtStar, retlOrders);
    }

    /**information sharing scenario
     * inf== 1 (INFORMATION_SHARING = 1)
     */

    else if(inf==1){
        System.err.println("with information sharing scenario!!");
        int dtstar1 = 0;

        if(w==1){
            dtStar.add(new Integer(
                ((Transaction)retailer.custOrders.elementAt(w-
1)).qty)
            );
            dtstar1 =
forecaster.computeDstar1(((Transaction)retailer.custOrders.elementAt(w-1)).qty,
                ((Integer)dtStar.elementAt(w-1)).intValue());
        } else if(w > 1){
            dtstar1 =
forecaster.computeDstar1(((Transaction)retailer.custOrders.elementAt(w-1)).qty,
                ((Integer)dtStar.elementAt(w-1)).intValue());
        }
    }

```

```

dtStar.add(new Integer(dtstar1));
System.out.println("DC-week: " + w+", forecasted demand for next week-
information sharing: "+dtstar1);

```

```

/** Compute qty for this order
 * @param dtstar1: forecasted retailer demand in week t+1
 * @param et: maximum inventory target
 *
 */

```

```

double otstar=0.0;
otstar = forecaster.computeOtstar(dtStar, retailer.custOrders);

```

```

}
else if(inf== 2){
System.err.println("with information sharing scenario new!!");
//
int dtstar1 = 0;

if(w==1){
dtStar.add(new Integer(
((Transaction)retlOrders.elementAt(w-1)).qty)
);
dtstar1 =
forecaster.computeDstar1(((Transaction)retlOrders.elementAt(w-1)).qty,
((Integer)dtStar.elementAt(w-1)).intValue());
} else if(w > 1){
dtstar1 = forecaster.computeDstar1(((Transaction)retlOrders.elementAt(w-
1)).qty,
((Integer)dtStar.elementAt(w-1)).intValue());
}
}

```

```

dtStar.add(new Integer(dtstar1));
System.out.println("DC-week: " + w+", forecasted demand for next week-
no inforamtion sharing: "+dtstar1);

```

```

//

```

```

/** Compute qty for this order
 * @param dtstar1: forecasted retailer demand in week t+1
 * @param et: maximum inventory target
 *
 */

```

```

        double otstar=0.0;
        otstar = forecaster.computeOtstar(dtStar, retailer.custOrders);

    }
    int qty = 0;
    int et = 0;
    et = forecaster.computeEt();
    System.out.println("et- weekly replenish inventory target is: "+et);
    qty = forecaster.computeQt(stockLevel.qty, backOrderQty_DC_MF);
    System.out.println("weekly submitted order from DC to MF: " + qty);

    //creat new order
    Transaction order = new Transaction(w, qty);
    //submit order
    manufacturer.takeOrder(order);
    //keep track of what have been ordered
    orders.add(order);
    System.out.println("DC-Week:"+w+", submittedOrder:"+order.toString());
}

public void insertMFSupplyIntoBuffer(int qty) {
    // TODO Auto-generated method stub
    bufferFromMF[bufferFromMF.length-1] = qty;
}

// manufacturer update the backorder list of DC
public void updateBackorder(int backOrderQty2) {

    // TODO Auto-generated method stub
    backOrderQty_DC_MF = backOrderQty2;
}

public void computeweeklyCost(int w){
    double totalCostQty; //weekly retailer total cost;
    if (((Transaction)orders.elementAt(w-1)).qty!= 0){
        totalCostQty= HOLDING_COST * ((Transaction)stockLevels.elementAt(w-
1)).qty
                                + ORDER_COST
                                + STOCKOUT_COST_PERUNIT *
((Transaction)backOrders_R_DC.elementAt(w-1)).qty;
    } else {
        totalCostQty = HOLDING_COST *
((Transaction)stockLevels.elementAt(w-1)).qty
                                + STOCKOUT_COST_PERUNIT *
((Transaction)backOrders_R_DC.elementAt(w-1)).qty;
    }
}

```

```

Transaction weeklyCost = new Transaction(w, totalCostQty);

weeklyCost.qy = totalCostQty;
weeklyCost.week = w;

weeklyCosts.add(weeklyCost);
System.out.println("Distributor weekly cost: " +
((Transaction)weeklyCosts.elementAt(w-1)).qy);
}

/** Compute the last n weeks retailer replenishment orders MEAN, MAD; and
 * the last n weeks retailer end inventory level MEAN, MAD; AND
 * the last n weeks retailer stockout MEAN, MAD.
 *
 * @param n PERIOD is used to compute the MEAN, MAD.
 */
public void computedcOrders_MEAN_MAD(int n){

    double sum = 0;
    for(int i= 1; i<=n; i++){
        sum += ((Transaction)orders.elementAt(orders.size()-i)).qty;
    }
    dcOrders_MEAN = 1.0*sum/n;

    sum = 0;
    for(int i = 1; i<=n; i++){
        sum += Math.abs((((Transaction)orders.elementAt(orders.size()-i)).qty)-
dcOrders_MEAN);
    }
    dcOrders_MAD = 1.0*sum/n;

    System.out.println("dcOrders_MEAN: "+dcOrders_MEAN+" dcOrders_MAD:
"+dcOrders_MAD);
}

public void computedcInventorylevel_MEAN_MAD(int n){

    double sum = 0;
    for(int i= 1; i<=n; i++){
        sum += ((Transaction)stockLevels.elementAt(stockLevels.size()-i)).qty;
    }
    dcInventorylevel_MEAN = 1.0*sum/n;

    sum = 0;
    for(int i = 1; i<=n; i++){

```

```

        sum +=
Math.abs((((Transaction)stockLevels.elementAt(stockLevels.size()-i)).qty)-
dcInventorylevel_MEAN);
    }
    dcInventorylevel_MAD = 1.0*sum/n;

    System.out.println("dcInvenotylevel_MEAN: "+dcInventorylevel_MEAN+"
dcInventorylevel_MAD: "+dcInventorylevel_MAD);
}

```

```

public void computedcBackorder_MEAN_MAD(int n){

    double sum = 0;
    for(int i= 1; i<=n; i++){
        sum +=
((Transaction)backOrders_R_DC.elementAt(backOrders_R_DC.size()-i)).qty;
    }
    dcBackorder_MEAN = 1.0*sum/n;

    sum = 0;
    for(int i = 1; i<=n; i++){
        sum +=
Math.abs((((Transaction)backOrders_R_DC.elementAt(backOrders_R_DC.size()-
i)).qty)-dcBackorder_MEAN);
    }
    dcBackorder_MAD = 1.0*sum/n;

    System.out.println("dcStockout_MEAN: "+dcBackorder_MEAN+"
dcStockout_MAD: "+dcBackorder_MAD);
}

```

```

public void computedcWeeklycost_MEAN_MAD(int n){

    double sum = 0;
    for(int i= 1; i<=n; i++){
        sum += ((Transaction)weeklyCosts.elementAt(weeklyCosts.size()-i)).qy;
    }
    dcWeeklycost_MEAN = 1.0*sum/n;

    sum = 0;
    for(int i = 1; i<=n; i++){
        sum +=
Math.abs((((Transaction)weeklyCosts.elementAt(weeklyCosts.size()-i)).qy)-
dcWeeklycost_MEAN);
    }
    dcWeeklycost_MAD = 1.0*sum/n;
}

```

```
        System.out.println("dcWeeklycost_MEAN: "+dcWeeklycost_MEAN+"  
dcWeeklycost_MAD: "+dcWeeklycost_MAD);  
    }  
}
```

## DistributorForecaster.java

```
import java.util.Vector;

/*
 * DistributorForecaster.java
 *
 * This class acts as the demand forecaster and inventory planning for DC.
 */

public class DistributorForecaster {

    double ORDER_COST;
    double UNIT_PURCHASE_COST;
    double HOLDING_COST;
    int LEAD_TIME_FROMMF;
    double STOCKOUT_COST_PERUNIT;
    double SAFETY_STOCK_PARAMETER;
    double ALPHA;

    double sc; //stockout or backorder cost
    int dstar1;
    int et;
    double tc;
    int qt; //replenishment order;
    double Otstar;

    /**
     * Creat a new instance of DCForecaster
     * @param o order cost
     * @param u unit purchase cost
     * @param h holding cost
     * @param l lead time from DC
     * @param s stockout cost per unit
     *
     */

    public DistributorForecaster(double o, double u, double h, int l, double s,double z,
double a){
        ORDER_COST = o;
        UNIT_PURCHASE_COST = u;
```

```

        HOLDING_COST = h;
        LEAD_TIME_FROMMF = l;
        STOCKOUT_COST_PERUNIT = s;
        SAFETY_STOCK_PARAMETER = z;
        ALPHA = a;
    }
}

/**
 * @param uiStar forecasted weekly customer demand
 * @param di      actual weekly customer demand
 */

public double computeOtstar( Vector uiStar, Vector di){
    int sum = 0;
    for(int i = 0; i < di.size() ; i++)
    {
        int q = ((Integer)uiStar.elementAt(i)).intValue()-
        ((Transaction)di.elementAt(i)).qty;
        sum = sum+ Math.abs(q);
    }
    Otstar = sum/di.size();

    System.out.println("Otstar from forecaster: " + Otstar);
    return(Otstar);
}

/**Compute Dstar1
 * @param dstar1 forecasted customer demand in week t+1;
 * @param dt      real customer demand in week t
 * @param dtstar  forecasted customer demand in week t. this value is
 *                the same value as dt in week 1.
 */

public int computeDstar1(int dt, int dtstar) {
    dstar1 = (int)Math.round(ALPHA*dt + (1-ALPHA)*dtstar);
    System.out.println("forecasted demand from forecaster:"+ dstar1);
    return dstar1;
}

/**
 * Compute et: maximum inventory target
 */

public int computeEt() {

    et = (int) (dstar1*(LEAD_TIME_FROMMF) +
    SAFETY_STOCK_PARAMETER* Math.sqrt(LEAD_TIME_FROMMF)*Otstar);
}

```

```

System.out.println("et-maximum inventory target from forecaster:"+et);
System.out.println("Otstar from forecaster: " + Otstar);
System.out.println("forecasted demand from forecaster:"+ dstar1);
return et;
}

/** Compute qt: replenishment order.
 * @param o on-hand inventory level
 * @param b back order from MF
 */
public int computeQt(int o, int b){
    qt = 0;
    if (et >= (o+b)){
        qt = et - (o+b);
    }
    System.out.println("DC replenishment order to MF from forecaster: "+ qt);
    return qt;
}
}

```

## Liner.java

```
import java.io.*;

public class Liner {
    String fname = null;
    BufferedReader reader = null;

    /**
     * Creates a new instance of Liner from a string
     * @param f      filename.
     */
    public Liner(String f) {
        fname = f;
        try {
            reader = new BufferedReader(new FileReader(fname));
        } catch(FileNotFoundException e) {
            e.printStackTrace();
        }
    }

    /**
     * Create a new instance from a BufferedReader.
     */
    public Liner(BufferedReader r) {
        reader = r;
    }

    /**
     * Set reader.
     */
    public void setReader(BufferedReader r) { reader = r; }

    public String readLine() {
        try {
            return reader.readLine();
        } catch(IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

```
public void close() {  
    try {  
        reader.close();  
    } catch(IOException e) {  
        e.printStackTrace();  
    }  
}  
  
}
```

## logger.java:

```
import java.io.*;

public class Logger {

    String fname = null;
    PrintWriter pw = null;
    /** Creates a new instance of Logger */
    public Logger(String f) {
        fname = f;
        try {
            pw = new PrintWriter(fname);
        } catch(IOException e) {
            e.printStackTrace();
        }
    }

    public void logFile(String s) {
        try {
            pw.write(s);
            pw.flush();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    public void log(String m) {
        logFile(m);
        logOut(m);
    }

    public static void logOut(String m) {
        System.out.print(m);
    }

    public static void logErr(String m) {
        System.err.print(m);
    }

    public void close() {
        pw.close();
    }
}
```

## Main.java:

```
import java.util.Vector;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class Main {
    Customer customer;
    Retailer retailer;
    Distributor distributor;
    Manufacturer manufacturer;
    Supplier supplier;
    int WEEKNO;
    int PERIOD;
    int CUSORDERMEAN;
    int CUSORDERMAD;
    int INFORMATIONSHARING;

    public void run(String pf) {
        Properties prop= new Properties();
        try{
            try {
                prop.load(new FileInputStream(new File(pf)));
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            WEEKNO = Integer.parseInt(prop.getProperty("WEEKNO"));
            PERIOD = Integer.parseInt(prop.getProperty("PERIOD"));
            CUSORDERMEAN
=Integer.parseInt(prop.getProperty("MEAN"));
            CUSORDERMAD = Integer.parseInt(prop.getProperty("MAD"));
            INFORMATIONSHARING =
Integer.parseInt(prop.getProperty("INFORMATIONSHARING"));

            } catch (NumberFormatException e){
                e.printStackTrace();
            }
        }
    }
}
```

```

for(int i=1; i<=WEEKNO; i++) {

    customer.submitOrder(i);
    retailer.receiveScheduledSupply(i);
    retailer.processOrders(i);

    retailer.submitOrder(i);
    retailer.computeweeklyCost(i);
    //if (i!=1 && i%52==0)
        //retailer.computeannualTotalCosts(i);

    distributor.receiveScheduledSupply(i);
    distributor.processOrders(i);

    distributor.submitOrder(i,INFORMATIONSHARING);
    distributor.computeweeklyCost(i);

    manufacturer.receiveScheduledSupply(i);
    manufacturer.processOrders(i);

    manufacturer.submitOrder(i,INFORMATIONSHARING);
    manufacturer.computeweeklyCost(i);

    supplier.processOrders(i);
}
retailer.computerelOrders_MEAN_MAD(PERIOD);
retailer.computerelInventorylevel_MEAN_MAD(PERIOD);
retailer.computerelStockout_MEAN_MAD(PERIOD);
retailer.computerelWeeklycost_MEAN_MAD(PERIOD);

distributor.computedcOrders_MEAN_MAD(PERIOD);
distributor.computedcInventorylevel_MEAN_MAD(PERIOD);
distributor.computedcBackorder_MEAN_MAD(PERIOD);
distributor.computedcWeeklycost_MEAN_MAD(PERIOD);

manufacturer.computemfOrders_MEAN_MAD(PERIOD);
manufacturer.computemfInventorylevel_MEAN_MAD(PERIOD);
manufacturer.computemfBackorder_MEAN_MAD(PERIOD);
manufacturer.computemfWeeklycost_MEAN_MAD(PERIOD);

System.out.println("Printing orders:");
// Print customer's orders in CSV (comma delimited)
    Logger ordersFile = new Logger("orders.csv");

```

```

Vector orders = customer.orders;
for(int i=0; i<orders.size(); i++)
    ordersFile.log(((Transaction)orders.elementAt(i)).qty+(i<orders.size()-1?" ":"\n"));
    //System.out.print(((Transaction)orders.elementAt(i)).qty+(i<orders.size()-
1?" ":"\n"));

    // Print retailer's orders in CSV (comma delimited)
orders = retailer.orders;
for(int i=0; i<orders.size(); i++)
    ordersFile.log(((Transaction)orders.elementAt(i)).qty+(i<orders.size()-1?" ":"\n"));
    //System.out.print(((Transaction)orders.elementAt(i)).qty+(i<orders.size()-
1?" ":"\n"));
    // Print DC's orders in CSV (comma delimited)
orders = distributor.orders;
for(int i=0; i<orders.size(); i++)
    ordersFile.log(((Transaction)orders.elementAt(i)).qty+(i<orders.size()-1?" ":"\n"));
    //System.out.print(((Transaction)orders.elementAt(i)).qty+(i<orders.size()-
1?" ":"\n"));
    // Print MF's orders in CSV (comma delimited)
orders = manufacturer.orders;
for(int i=0; i<orders.size(); i++)
    ordersFile.log(((Transaction)orders.elementAt(i)).qty+(i<orders.size()-1?" ":"\n"));
    //System.out.print(((Transaction)orders.elementAt(i)).qty+(i<orders.size()-
1?" ":"\n"));
ordersFile.close();

//Print out inventory levels after processing orders
System.out.println("Printing inventory after processing orders");
// Print customer's orders in CSV (comma delimited)

Logger inventoryFile = new Logger("inventorylevel.csv");
Vector stockLevels = retailer.stockLevels;
for(int i=0; i<stockLevels.size(); i++)

inventoryFile.log(((Transaction)stockLevels.elementAt(i)).qty+(i<stockLevels.size()-
1?" ":"\n"));

//System.out.print(((Transaction)stockLevels.elementAt(i)).qty+(i<stockLevels.size()-
1?" ":"\n"));
    // Print retailer's orders in CSV (comma delimited)
stockLevels = distributor.stockLevels;
for(int i=0; i<stockLevels.size(); i++)

inventoryFile.log(((Transaction)stockLevels.elementAt(i)).qty+(i<stockLevels.size()-
1?" ":"\n"));

```

```

//System.out.print(((Transaction)stockLevels.elementAt(i)).qty+(i<stockLevels.size()-
1?" ":"\n"));
    // Print DC's orders in CSV (comma delimited)
    stockLevels = manufacturer.stockLevels;
    for(int i=0; i<stockLevels.size(); i++)

inventoryFile.log(((Transaction)stockLevels.elementAt(i)).qty+(i<stockLevels.size()-
1?" ":"\n"));

//System.out.print(((Transaction)stockLevels.elementAt(i)).qty+(i<stockLevels.size()-
1?" ":"\n"));
    // Print MF's orders in CSV (comma delimited)
    inventoryFile.close();

    //print out backorder qtys

    System.out.println("Printing backorder qty");
    // Print customer's orders in CSV (comma delimited)

    Logger backOrderFile = new Logger("backorder.csv");
    Vector stockOuts = retailer.stockOuts;
    for(int i=0; i<stockOuts.size(); i++)

backOrderFile.log(((Transaction)stockOuts.elementAt(i)).qty+(i<stockOuts.size()-
1?" ":"\n"));
    //System.out.print(((Transaction)stockOuts.elementAt(i)).qty+(i<stockOuts.size()-
1?" ":"\n"));
    Vector backOrders = distributor.backOrders_R_DC;
    for(int i=0; i<backOrders.size(); i++)

backOrderFile.log(((Transaction)backOrders.elementAt(i)).qty+(i<stockOuts.size()-
1?" ":"\n"));

//System.out.print(((Transaction)backOrders.elementAt(i)).qty+(i<backOrders.size()-
1?" ":"\n"));
    backOrders = manufacturer.backOrders_DC_MF;
    for(int i=0; i<backOrders.size(); i++)

backOrderFile.log(((Transaction)backOrders.elementAt(i)).qty+(i<stockOuts.size()-
1?" ":"\n"));

//System.out.print(((Transaction)backOrders.elementAt(i)).qty+(i<backOrders.size()-
1?" ":"\n"));
    backOrderFile.close();

```

```

System.out.println("Printing weekly costs");
// Print weekly costs in CSV (comma delimited)

Logger weeklyCostsFile = new Logger("weeklyCosts.csv");
Vector weeklyCosts = retailer.weeklyCosts;
for(int i=0; i<weeklyCosts.size(); i++)

    weeklyCostsFile.log(((Transaction)weeklyCosts.elementAt(i)).qy+(i<weeklyCost
s.size()-1?" ":"\n"));
    weeklyCosts = distributor.weeklyCosts;
    for(int i=0; i<weeklyCosts.size(); i++)

        weeklyCostsFile.log(((Transaction)weeklyCosts.elementAt(i)).qy+(i<weeklyCost
s.size()-1?" ":"\n"));
        weeklyCosts = manufacturer.weeklyCosts;
        for(int i=0; i<weeklyCosts.size(); i++)

            weeklyCostsFile.log(((Transaction)weeklyCosts.elementAt(i)).qy+(i<weeklyCost
s.size()-1?" ":"\n"));
            weeklyCostsFile.close();

```

```

System.out.println("Printing MEAN MAD simulation analysis");
Logger MEAN_MAD_SimulationAnalysisFile = new Logger("MEAN_MAD.csv");
MEAN_MAD_SimulationAnalysisFile.log(CUSORDERMEAN+ ",");
MEAN_MAD_SimulationAnalysisFile.log(CUSORDERMAD+ "\n");
MEAN_MAD_SimulationAnalysisFile.log(retailer.relOrders_MEAN+ ",");
MEAN_MAD_SimulationAnalysisFile.log(retailer.relOrders_MAD+ ",");
MEAN_MAD_SimulationAnalysisFile.log(retailer.relInventorylevel_MEAN+ ",");
MEAN_MAD_SimulationAnalysisFile.log(retailer.relInventorylevel_MAD+ ",");
MEAN_MAD_SimulationAnalysisFile.log(retailer.relStockout_MEAN+ ",");
MEAN_MAD_SimulationAnalysisFile.log(retailer.relStockout_MAD+ ",");
MEAN_MAD_SimulationAnalysisFile.log(retailer.relWeeklycost_MEAN+ ",");
MEAN_MAD_SimulationAnalysisFile.log(retailer.relWeeklycost_MAD+ "\n");

MEAN_MAD_SimulationAnalysisFile.log(distributor.dcOrders_MEAN+ ",");
MEAN_MAD_SimulationAnalysisFile.log(distributor.dcOrders_MAD+ ",");
MEAN_MAD_SimulationAnalysisFile.log(distributor.dcInventorylevel_MEAN+
",");
MEAN_MAD_SimulationAnalysisFile.log(distributor.dcInventorylevel_MAD+ ",");
MEAN_MAD_SimulationAnalysisFile.log(distributor.dcBackorder_MEAN+ ",");
MEAN_MAD_SimulationAnalysisFile.log(distributor.dcBackorder_MAD+ ",");
MEAN_MAD_SimulationAnalysisFile.log(distributor.dcWeeklycost_MEAN+ ",");
MEAN_MAD_SimulationAnalysisFile.log(distributor.dcWeeklycost_MAD+ "\n");

```

```

        MEAN_MAD_SimulationAnalysisFile.log(manufacturer.mfOrders_MEAN+ "," );
        MEAN_MAD_SimulationAnalysisFile.log(manufacturer.mfOrders_MAD+ "," );

        MEAN_MAD_SimulationAnalysisFile.log(manufacturer.mfInventorylevel_MEAN+ "," );
        MEAN_MAD_SimulationAnalysisFile.log(manufacturer.mfInventorylevel_MAD+
        "," );
        MEAN_MAD_SimulationAnalysisFile.log(manufacturer.mfBackorder_MEAN+
        "," );
        MEAN_MAD_SimulationAnalysisFile.log(manufacturer.mfBackorder_MAD+ "," );
        MEAN_MAD_SimulationAnalysisFile.log(manufacturer.mfWeeklycost_MEAN+
        "," );
        MEAN_MAD_SimulationAnalysisFile.log(manufacturer.mfWeeklycost_MAD+
        "\n");

```

```

        MEAN_MAD_SimulationAnalysisFile.close();

```

```

    }

```

```

    public Main(String orderFile) {
        Liner liner = new Liner(orderFile);
        String [] data = liner.readLine().split(",");
        liner.close();
        int [] orderData = new int [data.length];
        for(int i=0; i<data.length; i++)
            orderData[i] = Integer.parseInt(data[i]);
        customer = new Customer(orderData);
        retailer = new Retailer("retailer.properties");
        distributor = new Distributor("distributor.properties");

        manufacturer = new Manufacturer("manufacturer.properties");
        supplier = new Supplier();

        customer.setRetailer(retailer);
        retailer.setCustomer(customer);
        retailer.setDistributor(distributor);

        distributor.setRetailer(retailer);
        distributor.setManufactuer(manufacturer);
        manufacturer.setRetailer(retailer);
        manufacturer.setDistributor(distributor);
        manufacturer.setSupplier(supplier);
    }

```

```

supplier.setManufacturer(manufacturer);
}
/**
 * @param args
 */
public static void main(String[] args) {
    // TODO Auto-generated method stub
    System.out.println("Hello, this is the supply chain simulation project!");
    if(args.length!=1){
        System.out.println("Usage: Main CustomerOrder.csv");
    }
    new Main(args[0]).run("Simulation parameter.properties");
}
}

```

## Manufacturer.java:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;
import java.util.Vector;

public class Manufacturer {

    /** Creates a new instance of Manufacturer */
    Vector dcOrders;      //dt real distributor demand in week t.
    Vector dtStar; //forecasted DC demand in week t.
    Vector orders;
    Vector stockLevels;
    Vector backOrders_DC_MF;
    Vector weeklyCosts;

    double mfOrders_MEAN;
    double mfOrders_MAD;
    double mfInventorylevel_MEAN;
    double mfInventorylevel_MAD;
    double mfBackorder_MEAN;
    double mfBackorder_MAD;
    double mfWeeklycost_MEAN;
    double mfWeeklycost_MAD;

    Transaction weeklyCost;
    int backOrderQty_DC_MF; //updated backorder quantity between DC and MF
    Transaction submittedOrder;
    Distributor distributor;
    Retailer retailer;
    Supplier supplier;
    int qtyReceivedFromSupplier;
    Transaction stockLevel;
    int[] bufferFromSP; //the size of buffer is leadtimeofMF+1

    String propFile;
    ManufacturerForecaster forecaster;
```

```

double ORDER_COST;
double UNIT_PURCHASE_COST;
double UNIT_PRODUCTION_COST;
double HOLDING_COST;
int LEAD_TIME_FROMSP;
double STOCKOUT_COST_PERUNIT;
int INITIAL_INVENTORY_LEVEL;
double SAFETY_STOCK_PARAMETER;
double ALPHA;

public Manufacturer(String pf) {
    Properties prop = new Properties();
    try{
        prop.load(new FileInputStream(new File(pf)));

        ORDER_COST =
Double.parseDouble(prop.getProperty("ORDER_COST"));
        UNIT_PURCHASE_COST =
Double.parseDouble(prop.getProperty("UNIT_PURCHASE_COST"));
        UNIT_PRODUCTION_COST =
Double.parseDouble(prop.getProperty("UNIT_PRODUCTION_COST"));
        HOLDING_COST =
Double.parseDouble(prop.getProperty("HOLDING_COST"));
        LEAD_TIME_FROMSP =
Integer.parseInt(prop.getProperty("LEAD_TIME_FROMSP"));
        STOCKOUT_COST_PERUNIT =
Double.parseDouble(prop.getProperty("STOCK_COST"));
        INITIAL_INVENTORY_LEVEL =
Integer.parseInt(prop.getProperty("INITIAL_INVENTORY_LEVEL"));
        SAFETY_STOCK_PARAMETER =
Double.parseDouble(prop.getProperty("SAFETY_STOCK_PARAMETER"));
        ALPHA = Double.parseDouble(prop.getProperty("ALPHA"));
        forecaster = new ManufacturerForecaster(ORDER_COST,
UNIT_PURCHASE_COST, HOLDING_COST, LEAD_TIME_FROMSP,
STOCKOUT_COST_PERUNIT, SAFETY_STOCK_PARAMETER, ALPHA);

    } catch(FileNotFoundException e){
        e.printStackTrace();
    } catch(IOException e){
        e.printStackTrace();
    } catch(NumberFormatException e){
        e.printStackTrace();
    }
}

dcOrders = new Vector();
dtStar = new Vector();

```

```

orders = new Vector();
stockLevels = new Vector();
backOrders_DC_MF = new Vector();
weeklyCosts = new Vector();

submittedOrder = null;
distributor = null;
supplier = null;
qtyReceivedFromSupplier = 0;
stockLevel = null;
bufferFromSP = new int[LEAD_TIME_FROMSP+1];
backOrderQty_DC_MF = 0;
propFile = pf;

mfOrders_MEAN = 0;
mfOrders_MAD=0;
mfInventorylevel_MEAN=0;
mfInventorylevel_MAD=0;
mfBackorder_MEAN=0;
mfBackorder_MAD=0;
mfWeeklycost_MEAN=0;
mfWeeklycost_MAD=0;

}

public void setDistributor(Distributor r) { distributor = r; }

public void setSupplier(Supplier s) { supplier = s; }

public void setRetailer(Retailer r) {retailer = r;}

public void takeOrder(Transaction o) {
submittedOrder = new Transaction(o.week, o.qty);
dcOrders.add(submittedOrder);
}
// receive scheduled supply from Supplier and update stock level
public void receiveScheduledSupply(int w) {
// take the front element from the buffer
int qtyReceivedFromSupplier = bufferFromSP[0];
//update the buffer push qty forward everyweel
for (int i=0; i<bufferFromSP.length-1; i++){
bufferFromSP[i] = bufferFromSP[i+1];
}
bufferFromSP[bufferFromSP.length-1]=0;

```

```

        // initialize current level in stock
        int level = 0;
        // carry out the current stock level with last week's one
        if(w==1) {
            // Random initial value of level between 50-100
            //level = ((int)(Math.random()*50)+50);

            //Get the initial inventory level from the user
            level = INITIAL_INVENTORY_LEVEL;
        } else {
            level = ((Transaction)stockLevels.elementAt(w-2)).qty;
        }
        System.out.println("MF-Week:"+w+",initial Level from last
week:"+level+
                        ", qty of supply from
Supplier:"+qtyReceivedFromSupplier);
        // update the current stock level with qty received from MF
        stockLevel = new Transaction(w, level+qtyReceivedFromSupplier);
        System.out.println("MF-Week:"+w+", updated stockLevel BEFORE
processing order:"+stockLevel.toString());
    }

    // process order received from Manufacturer
    public void processOrders(int w) {
        if(submittedOrder.qty + backOrderQty_DC_MF >=stockLevel.qty) {
            // update distributor's buffer
            distributor.insertMFSupplyIntoBuffer(stockLevel.qty);
            System.out.println("MF->=, submittedOrder.qty:"+submittedOrder.qty+
                            ", stockLevel.qty"+stockLevel.qty);
            backOrderQty_DC_MF = submittedOrder.qty + backOrderQty_DC_MF -
stockLevel.qty;
            stockLevel.qty = 0;

        } else {
            // update distributor's buffer

distributor.insertMFSupplyIntoBuffer(submittedOrder.qty+backOrderQty_DC_MF);
            System.out.println("MF-<, submittedOrder.qty:"+submittedOrder.qty+
                            ", stockLevel.qty"+stockLevel.qty);
            stockLevel.qty -= submittedOrder.qty;
            backOrderQty_DC_MF = 0;
        }

        // inform DC about the backorder of the order
        distributor.updateBackorder(backOrderQty_DC_MF);
        System.out.println("MF backOrder to DC:"+backOrderQty_DC_MF);
    }

```

```

        // update MF's stockLevels
        stockLevels.add(stockLevel);
        System.out.println("MF-Week:"+w+", updated stockLevel AFTER processing
order:"+stockLevel.toString());

        //record manufacturer's backorder level (between DC & MF)
        Transaction backOrder = new Transaction(w, backOrderQty_DC_MF);
        backOrders_DC_MF.add(backOrder);

    }

    // distributor insert it's supply into retailer's buffer
    public void insertSPSupplyIntoBuffer(int q) {
        bufferFromSP[bufferFromSP.length-1] = q;
    }

    //submit order to SP (EVERY TSTAR WEEKS)
    public void submitOrder(int w, int info) {

        System.err.println("ALPHA: "+ ALPHA + "Safety stock parameter: " +
SAFETY_STOCK_PARAMETER);
        /**No information sharing scenario
        * info == 0: INFORMATIONSHARING ==0
        */
        if(info == 0){
            int dtstar1 = 0;
            if(w==1) {
                dtStar.add(new Integer(((Transaction)dcOrders.elementAt(w-
1)).qty));
                dtstar1 =

                forecaster.computeDstar1(((Transaction)dcOrders.elementAt(w-1)).qty,
                ((Integer)dtStar.elementAt(w-1)).intValue());
            }

            else if(w>1) {
                dtstar1 =

                forecaster.computeDstar1(((Transaction)dcOrders.elementAt(w-1)).qty,
                ((Integer)dtStar.elementAt(w-1)).intValue());
            }
        }
    }

```

```

        dtStar.add(new Integer(dtstar1));
        System.out.println("MF-Week:" + w+", forecasted demand for next week:
" +dtstar1);

        double otstar = 0.0;
        otstar = forecaster.computeOtstar(dtStar, dcOrders);

    }

    /**With information sharing scenario
    * info == 1: INFORMATIONSHARING ==1
    */
    else if(info == 1){
        int dtstar1 = 0;
        if(w==1) {
            dtStar.add(new
Integer(((Transaction)retailer.custOrders.elementAt(w-1)).qty));
            dtstar1 =

                forecaster.computeDstar1(((Transaction)retailer.custOrders.elementAt(w-1)).qty,
((Integer)dtStar.elementAt(w-1)).intValue());
        }

        else if(w>1) {
            dtstar1 =

                forecaster.computeDstar1(((Transaction)retailer.custOrders.elementAt(w-1)).qty,
((Integer)dtStar.elementAt(w-1)).intValue());

        }

        dtStar.add(new Integer(dtstar1));
        System.out.println("MF-Week:" + w+", forecasted demand for next week:
" +dtstar1);

        double otstar = 0.0;
        otstar = forecaster.computeOtstar(dtStar, retailer.custOrders);

    }
    else if(info == 2){
        int dtstar1 = 0;

```

```

        if(w==1) {
            dtStar.add(new Integer(((Transaction)dcOrders.elementAt(w-
1)).qty));
            dtstar1 =

            forecaster.computeDstar1(((Transaction)dcOrders.elementAt(w-1)).qty,
((Integer)dtStar.elementAt(w-1)).intValue());
        }

        else if(w>1) {
            dtstar1 =

            forecaster.computeDstar1(((Transaction)dcOrders.elementAt(w-1)).qty,
((Integer)dtStar.elementAt(w-1)).intValue());

        }

        dtStar.add(new Integer(dtstar1));
        System.out.println("MF-Week:" + w+", forecasted demand for next week:
" +dtstar1);

        double otstar = 0.0;
        otstar = forecaster.computeOtstar(dtStar, retailer.custOrders);

    }
    else{ System.err.println("erro: INFORMATIONSAHRING PARAMETER
ONLY CAN BE 0 OR 1");}

    int qty = 0;
    int et=0;

    et = forecaster.computeEt();
    System.out.println("et - weekly replenish inventory target is: " + et);

    qty = forecaster.computeQt(stockLevel.qty);
    System.out.println("weekly submitted order from MF to SP: " + qty);

    //create new order
    Transaction order = new Transaction(w, qty);
    //submit order
    supplier.takeOrder(order);
    //keep track of what have been ordered
    orders.add(order);

```

```

        System.out.println("MF-week: " + w+", forecasted demand:" +dtStar.toString());
    }

    public void computeweeklyCost(int w){
        double totalCostQty;
        if (((Transaction)orders.elementAt(w-1)).qty!= 0){
            totalCostQty= HOLDING_COST * ((Transaction)stockLevels.elementAt(w-
1)).qty
                        + ORDER_COST
                        + STOCKOUT_COST_PERUNIT *
((Transaction)backOrders_DC_MF.elementAt(w-1)).qty
                        + (UNIT_PURCHASE_COST +
UNIT_PRODUCTION_COST)*(((Transaction)orders.elementAt(w-1)).qty);

        } else {
            totalCostQty = HOLDING_COST *
((Transaction)stockLevels.elementAt(w-1)).qty
                        + STOCKOUT_COST_PERUNIT *
((Transaction)backOrders_DC_MF.elementAt(w-1)).qty
                        +
(UNIT_PURCHASE_COST+UNIT_PRODUCTION_COST)*(((Transaction)orders.ele
mentAt(w-1)).qty);
        }
        Transaction weeklyCost = new Transaction(w, totalCostQty);

        weeklyCost.qy = totalCostQty;
        weeklyCost.week = w;

        weeklyCosts.add(weeklyCost);
        System.out.println("Manufacturer weekly cost: " +
((Transaction)weeklyCosts.elementAt(w-1)).qy);
    }

    /** Compute the last n weeks retailer replenishment orders MEAN, MAD; and
    * the last n weeks retailer end inventory level MEAN, MAD; AND
    * the last n weeks retailer stockout MEAN, MAD.
    *
    * @param      n      PERIOD      is used to compute the MEAN, MAD.
    */
    public void computemfOrders_MEAN_MAD(int n){

        double sum = 0;
        for(int i= 1; i<=n; i++){
            sum += ((Transaction)orders.elementAt(orders.size()-i)).qty;

```

```

    }
    mfOrders_MEAN = 1.0*sum/n;

    sum = 0;
    for(int i = 1; i<=n; i++){
        sum += Math.abs((((Transaction)orders.elementAt(orders.size()-i)).qty)-
mfOrders_MEAN);
    }
    mfOrders_MAD = 1.0*sum/n;

    System.out.println("mfOrders_MEAN: "+mfOrders_MEAN+" mfOrders_MAD:
"+mfOrders_MAD);
}

public void computemfInventorylevel_MEAN_MAD(int n){

    double sum = 0;
    for(int i= 1; i<=n; i++){
        sum += ((Transaction)stockLevels.elementAt(stockLevels.size()-i)).qty;
    }
    mfInventorylevel_MEAN = 1.0*sum/n;

    sum = 0;
    for(int i = 1; i<=n; i++){
        sum +=
Math.abs((((Transaction)stockLevels.elementAt(stockLevels.size()-i)).qty)-
mfInventorylevel_MEAN);
    }
    mfInventorylevel_MAD = 1.0*sum/n;

    System.out.println("mfInvenotylevel_MEAN: "+mfInventorylevel_MEAN+"
mfInventorylevel_MAD: "+ mfInventorylevel_MAD);
}

public void computemfBackorder_MEAN_MAD(int n){

    double sum = 0;
    for(int i= 1; i<=n; i++){
        sum +=
((Transaction)backOrders_DC_MF.elementAt(backOrders_DC_MF.size()-i)).qty;
    }
    mfBackorder_MEAN = 1.0*sum/n;

    sum = 0;
    for(int i = 1; i<=n; i++){

```

```

        sum +=
Math.abs((((Transaction)backOrders_DC_MF.elementAt(backOrders_DC_MF.size()-
i)).qty)-mfBackorder_MEAN);
    }
    mfBackorder_MAD = 1.0*sum/n;

    System.out.println("mfStockout_MEAN: "+mfBackorder_MEAN+"
mfStockout_MAD: "+ mfBackorder_MAD);
}

    public void computemfWeeklycost_MEAN_MAD(int n) {
        double sum = 0;
        for(int i= 1; i<=n; i++){
            sum +=
((Transaction)weeklyCosts.elementAt(weeklyCosts.size()-i)).qy;
        }
        mfWeeklycost_MEAN = 1.0*sum/n;

        sum = 0;
        for(int i = 1; i<=n; i++){
            sum +=
Math.abs((((Transaction)weeklyCosts.elementAt(weeklyCosts.size()-i)).qy)-
mfWeeklycost_MEAN);
        }
        mfWeeklycost_MAD = 1.0*sum/n;

        System.out.println("mfWeeklycost_MEAN: "+
mfWeeklycost_MEAN+" mfWeeklycost_MAD: "+ mfWeeklycost_MAD);

    }
}

```

## ManufacturerForecaster.java:

```
import java.util.Vector;

/*
 * ManufacturerForecaster.java
 *
 * This class acts as the demand forecaster and inventory planning for MF.
 */

public class ManufacturerForecaster {

    double ORDER_COST;
    double UNIT_PURCHASE_COST;
    double HOLDING_COST;
    int LEAD_TIME_FROMSP;
    double STOCKOUT_COST_PERUNIT;
    double SAFETY_STOCK_PARAMETER;
    double ALPHA;

    double sc; //stockout or backorder cost
    int dstar1;
    int et;
    double tc;
    int qt; //replenishment order;
    double Otstar;

    /**
     * Creat a new instance of MFForecaster
     * @param o order cost
     * @param u unit purchase cost
     * @param h holding cost
     * @param l lead time from DC
     * @param s stockout cost per unit
     *
     */

    public ManufacturerForecaster(double o, double u, double h, int l, double
s,double z, double a){
        ORDER_COST = o;
        UNIT_PURCHASE_COST = u;
        HOLDING_COST = h;
    }
}
```

```

        LEAD_TIME_FROMSP = 1;
        STOCKOUT_COST_PERUNIT = s;
        SAFETY_STOCK_PARAMETER = z;
        ALPHA = a;
    }

    /**Compute Otstar
    * @param uniStar forecasted weekly customer demand
    * @param di      actual weekly customer demand
    */

    public double computeOtstar(Vector uiStar, Vector di){
        int sum = 0;
        for(int i = 0; i < di.size(); i++)
        {
            int q = ((Integer)uiStar.elementAt(i)).intValue()-
((Transaction)di.elementAt(i)).qty;
            sum = sum + Math.abs(q);
        }

        Otstar = sum/di.size();
        System.out.println("Otstar from forecaster:" + Otstar);
        return (Otstar);
    }

    /**Compute Dstar1
    * @param dstar1 forecasted customer demand in week t+1;
    * @param dt      real customer demand in week t
    * @param dtstar  forecasted customer demand in week t. this value is
    *                the same value as dt in week 1.
    */

    public int computeDstar1(int dt, int dtstar){
        dstar1 = (int)Math.round(ALPHA*dt + (1-ALPHA)* dtstar);
        System.out.println("forecasted demand from forecaster:" + dstar1);
        return dstar1;
    }

    /**
    * Compute et: maximum inventory target
    */
    public int computeEt(){

```

```

        et =
(int)(dstar1*(LEAD_TIME_FROMSP)+SAFETY_STOCK_PARAMETER*Math.sqrt(L
EAD_TIME_FROMSP)*Otstar);

        System.out.println("et-maximum inventory target from forecaster: "+et);
        System.out.println("Otstar from forecaster"+Otstar);
        System.out.println("forecasted demand from forecaster: "+ dstar1);
        return et;
    }

    /** Compute qt: replenishment order.
    * @param o on-hand inventory level
    * @param b back order from SP
    *
    */
    public int computeQt(int o){
        qt = 0;
        if (et >= o){
            qt = et - o;
        }
        System.out.println("MF replenishment order to SP from forecaster: "+ qt);
        return qt;
    }
}

```

## Retailer.java:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;
import java.util.Vector;

public class Retailer {
    Vector custOrders; // dt real customer demand in week t
    Vector dtStar; // dtstar forecasted customer demand in week t.
    // This value is the same value as dt in week 1.
    Vector orders; // Replenishment orders from Retailer to DC
    Vector stockLevels; // current (updated) Inventory level
    Vector stockOuts;
    Vector weeklyCosts;

    double relOrders_MEAN; //the MEAN of replenishment orders from retailer to
DC
    double relOrders_MAD; //the MAD of replenishment orders from retailer to DC
    double relInventorylevel_MEAN;
    double relInventorylevel_MAD;
    double relStockout_MEAN;
    double relStockout_MAD;
    double relWeeklycost_MEAN;
    double relWeeklycost_MAD;

    Transaction weeklyCost;
    int backOrderQty_R_DC; //updated backorder quantity between DC and Retailer
    Transaction submittedOrder;
    Customer customer;
    Distributor distributor;
    Manufacturer manufacturer;
    int qtyReceivedFromDC;
    Transaction stockLevel;
    int [] bufferFromDC; // the size of buffer is LEAD_TIME_FROMDC + 1
    String propFile;
    RetailerForecaster forecaster;

    double ORDER_COST; //order cost per order
    double UNIT_PURCHASE_COST; //Unit purchase cost
```

```

double HOLDING_COST; //Holding cost per unit per year
int LEAD_TIME_FROMDC;
double STOCKOUT_COST_PERUNIT;
int INITIAL_INVENTORY_LEVEL; //Initial inventory level in week 1 (input from
the User)
double SAFETY_STOCK_PARAMETER; //z
double ALPHA;

public Retailer (String pf) {
    Properties prop = new Properties();
    try {
        prop.load(new FileInputStream(new File(pf)));

        ORDER_COST = Double.parseDouble(prop.getProperty("ORDER_COST"));
        UNIT_PURCHASE_COST =
Double.parseDouble(prop.getProperty("UNIT_PURCHASE_COST"));
        HOLDING_COST =
Double.parseDouble(prop.getProperty("HOLDING_COST"));
        LEAD_TIME_FROMDC =
Integer.parseInt(prop.getProperty("LEAD_TIME_FROMDC"));

        STOCKOUT_COST_PERUNIT =
Double.parseDouble(prop.getProperty("STOCK_COST"));
        INITIAL_INVENTORY_LEVEL =
Integer.parseInt(prop.getProperty("INITIAL_INVENTORY_LEVEL"));
        ALPHA = Double.parseDouble(prop.getProperty("ALPHA"));
        SAFETY_STOCK_PARAMETER =
Double.parseDouble(prop.getProperty("SAFETY_STOCK_PARAMETER"));

        forecaster = new RetailerForecaster(ORDER_COST, UNIT_PURCHASE_COST,
HOLDING_COST,
            LEAD_TIME_FROMDC,
STOCKOUT_COST_PERUNIT,SAFETY_STOCK_PARAMETER, ALPHA);

    } catch(FileNotFoundException e){
        e.printStackTrace();
    } catch(IOException e) {
        e.printStackTrace();
    } catch (NumberFormatException e) {
        e.printStackTrace();
    }
}

custOrders = new Vector();
    dtStar = new Vector();
    orders = new Vector();
    stockLevels = new Vector();

```

```

        stockOuts = new Vector();
        weeklyCosts = new Vector();

        submittedOrder = null;
        customer = null;
        qtyReceivedFromDC = 0;
        stockLevel = null;
        bufferFromDC = new int [LEAD_TIME_FROMDC+1];
        backOrderQty_R_DC = 0;
        propFile = pf;

        relOrders_MEAN = 0; //the MEAN of replenishment orders from retailer to DC
        relOrders_MAD = 0; //the MAD of replenishment orders from retailer to DC
        relInventorylevel_MEAN = 0;
        relInventorylevel_MAD = 0;
        relStockout_MEAN = 0;
        relStockout_MAD = 0;
        relWeeklycost_MEAN = 0;
        relWeeklycost_MAD = 0;

    }

    public void setCustomer(Customer c) { customer = c; }

    public void setDistributor(Distributor d) { distributor = d; }

    public void setManufacturer(Manufacturer m){manufacturer = m;}

    /**
     * Take order from customer
     */
    public void takeOrder(Transaction o) {
        submittedOrder = new Transaction(o.week, o.qty);
        custOrders.add(submittedOrder);

        System.err.println("new customer's order added: size="+custOrders.size());

    }
    // receive scheduled supply from DC and update stock level
    public void receiveScheduledSupply(int w) {
        // take the front element from the buffer
        int qtyReceivedFromDC = bufferFromDC[0];
        // update the buffer push qty forward every week
        for(int i=0; i<bufferFromDC.length-1; i++) {
            bufferFromDC[i] = bufferFromDC[i+1];

```

```

    }
    bufferFromDC[bufferFromDC.length-1] = 0;

    // initialize current level in stock
    int level = 0;
    // carry out the current stock level with last week's one
    if(w==1) {
        // get the initial inventory level from the User
        level = INITIAL_INVENTORY_LEVEL;
    } else {
        level = ((Transaction)stockLevels.elementAt(w-2)).qty;
    }
    System.out.println("RT-Week:"+w+",initial Level from last week:
"+level+
        ", qty of supply from DC: "+qtyReceivedFromDC);
    // update the current stock level with qty received from DC
    stockLevel = new Transaction(w, level+qtyReceivedFromDC);
    System.out.println("RT-Week: "+w+", updated stockLevel BEFORE
processing order: "+stockLevel.toString());
    }

    // process order submitted by customer
    public void processOrders(int w) {
        int outLevel = 0; // the level of stockout
        if(submittedOrder.qty>=stockLevel.qty) {
            System.out.println("RT->=, submittedOrder.qty:
"+submittedOrder.qty+
                ", stockLevel.qty: "+stockLevel.qty);
            outLevel = submittedOrder.qty-stockLevel.qty;
            stockLevel.qty = 0;
        } else {
            System.out.println("RT-<, submittedOrder.qty:
"+submittedOrder.qty+
                ", stockLevel.qty: "+stockLevel.qty);
            stockLevel.qty -= submittedOrder.qty;
            outLevel = 0;
        }
    }

    // update retailer's stockLevels
    stockLevels.add(stockLevel);
    System.out.println("RT-Week:"+w+", updated stockLevel AFTER
processing order: "+stockLevel.toString());

    // inform customer about the stockout of the order
    Transaction stockOut = new Transaction(w, outLevel);

```

```

        //customer.updateStockout(stockOut);

        // update retailer's stockOuts
        stockOuts.add(stockOut);
        System.out.println("RT-Week:"+w+", updated stockOut AFTER
processing order: "+stockOut.toString()+"\n");

    }

    // submit order to Distributor (every Tstar weeks)
    public void submitOrder(int w) {

        // Setup dtStar (forecasted customer demand in week t) for week 1 to be customer
order of week 1
        System.err.println("Alpha: "+ALPHA + "safety stock parameter:
"+SAFETY_STOCK_PARAMETER);
        int dtstar1= 0;

        if(w==1) {
            dtStar.add(new Integer(
                ((Transaction)custOrders.elementAt(w-1)).qty)
            );
            dtstar1 =
forecaster.computeDstar1(((Transaction)custOrders.elementAt(w-1)).qty,
                ((Integer)dtStar.elementAt(w-1)).intValue());
            //dtstar1 = ((Transaction)custOrders.elementAt(w-1)).qty;
        }

        /** Compute qty for this order
         * @param dtstar1: foresated customer demand in week t+1
         * @param et: maximum inventory target
         */

        else if(w > 1){
            dtstar1 =
forecaster.computeDstar1(((Transaction)custOrders.elementAt(w-1)).qty,
                ((Integer)dtStar.elementAt(w-1)).intValue());
        }

        dtStar.add(new Integer(dtstar1));
        System.out.println("RT-Week:"+w+", forecasted demand for next week: "+
dtstar1);

        int qty = 0;
        int et = 0;

```

```

double otstar=0.0;
otstar = forecaster.computeOtstar(dtStar, custOrders);
//if(w==1 || (w-1)% tStar_R==0){
    et = forecaster.computeEt();
    System.out.println("et- weekly replenish inventory target is: "+et);
    qty = forecaster.computeQt(stockLevel.qty, backOrderQty_R_DC);
    System.out.println("weekly submitted order from retailer to DC: " + qty);
// }

    // create new order
    Transaction order = new Transaction(w, qty);
    // submit order
    distributor.takeOrder(order);
    // keep track of what have been ordered
    orders.add(order);
    System.out.println("RT-Week:"+w+", submittedOrder: "+order.qty);
    // System.out.println("Printing tstar: " + tStar_R+",          alpha: "+ aLpha);
    System.out.println("RT-Week:"+w+", forecasted demand:"+dtStar.toString());
}

// distributor insert it's supply into retailer's buffer
public void insertDCSupplyIntoBuffer(int q) {
    bufferFromDC[bufferFromDC.length-1] = q;
}

// distributor update the backorder list of retailer
public void updateBackorder(int q) {
    backOrderQty_R_DC = q;
}

public void computeweeklyCost(int w){
    double totalCostQty; //weekly retailer total cost;
    if (((Transaction)orders.elementAt(w-1)).qty!= 0){
        totalCostQty= HOLDING_COST * ((Transaction)stockLevels.elementAt(w-
1)).qty
                                + ORDER_COST
                                + STOCKOUT_COST_PERUNIT *
((Transaction)stockOuts.elementAt(w-1)).qty;

    } else {
        totalCostQty = HOLDING_COST *
((Transaction)stockLevels.elementAt(w-1)).qty
                    + STOCKOUT_COST_PERUNIT *
((Transaction)stockOuts.elementAt(w-1)).qty;
    }
}

```

```

    }
    Transaction weeklyCost = new Transaction(w, totalCostQty);

    weeklyCost.qy = totalCostQty;
    weeklyCost.week = w;

    weeklyCosts.add(weeklyCost);
    System.out.println("Retailer weekly cost: " +
        ((Transaction)weeklyCosts.elementAt(w-1)).qy);
    }

    /** Compute the last n weeks retailer replenishment orders MEAN, MAD; and
     * the last n weeks retailer end inventory level MEAN, MAD; AND
     * the last n weeks retailer stockout MEAN, MAD.
     *
     * @param      n      PERIOD      is used to compute the MEAN, MAD.
     */
    public void computerelOrders_MEAN_MAD(int n){

        double sum = 0;
        for(int i= 1; i<=n; i++){
            sum += ((Transaction)orders.elementAt(orders.size()-i)).qty;
        }
        relOrders_MEAN = 1.0*sum/n;

        sum = 0;
        for(int i = 1; i<=n; i++){
            sum += Math.abs((((Transaction)orders.elementAt(orders.size()-i)).qty)-
relOrders_MEAN);
        }
        relOrders_MAD = 1.0*sum/n;

        System.out.println("relOrders_MEAN: "+relOrders_MEAN+" relOrders_MAD:
"+relOrders_MAD);
    }

    public void computerelInventorylevel_MEAN_MAD(int n){

        double sum = 0;
        for(int i= 1; i<=n; i++){
            sum += ((Transaction)stockLevels.elementAt(stockLevels.size()-i)).qty;
        }
        relInventorylevel_MEAN = 1.0*sum/n;

        sum = 0;
        for(int i = 1; i<=n; i++){

```

```

        sum +=
Math.abs((((Transaction)stockLevels.elementAt(stockLevels.size()-i)).qty)-
relInventorylevel_MEAN);
    }
    relInventorylevel_MAD = 1.0*sum/n;

    System.out.println("relInvenotylevel_MEAN: "+relInventorylevel_MEAN+"
relInventorylevel_MAD: "+relInventorylevel_MAD);
    }

    public void computerelStockout_MEAN_MAD(int n){

        double sum = 0;
        for(int i= 1; i<=n; i++){
            sum += ((Transaction)stockOuts.elementAt(stockOuts.size()-i)).qty;
        }
        relStockout_MEAN = 1.0*sum/n;

        sum = 0;
        for(int i = 1; i<=n; i++){
            sum += Math.abs((((Transaction)stockOuts.elementAt(stockOuts.size()-
i)).qty)-relStockout_MEAN);
        }
        relStockout_MAD = 1.0*sum/n;

        System.out.println("relStockout_MEAN: "+relStockout_MEAN+"
relStockout_MAD: "+relStockout_MAD);
    }

    public void computerelWeeklycost_MEAN_MAD(int n){

        double sum = 0;
        for(int i= 1; i<=n; i++){
            sum += ((Transaction)weeklyCosts.elementAt(weeklyCosts.size()-i)).qy;
        }
        relWeeklycost_MEAN = 1.0*sum/n;

        sum = 0;
        for(int i = 1; i<=n; i++){
            sum +=
Math.abs((((Transaction)weeklyCosts.elementAt(weeklyCosts.size()-i)).qy)-
relWeeklycost_MEAN);
        }
        relWeeklycost_MAD = 1.0*sum/n;
    }

```

```
        System.out.println("relWeeklycost_MEAN: "+relWeeklycost_MEAN+"  
relWeeklycost_MAD: "+relWeeklycost_MAD);  
    }  
  
}
```

## RetailForecaster.java:

```
import java.util.Vector;

/*
 * RetailForecaster.java
 *
 * This class acts as the demand forecaster and inventory planning for retailer.
 */

public class RetailForecaster {

    double ORDER_COST; // Order cost per order
    double UNIT_PURCHASE_COST; // Unit purchase cost
    double HOLDING_COST; // Holding cost per unit per year
    int LEAD_TIME_FROMDC; // Leadtime OF DC
    double STOCKOUT_COST_PERUNIT; // Stock cost per unit
    double SAFETY_STOCK_PARAMETER; //z
    double ALPHA;

    double sc; // Stockout cost
    int dstar1; // forecated week t+1 demand in units
    int et; // weekly maximum inventory target
    double tc; // weekly total cost
    int qt; // replenishment order
    double Otstar;
    double retailerAnnualTotoalCost; //annual total cost of retailer

    /**
     * Creates a new instance of RetailForecaster
     *
     * @param o    order cost
     * @param u    unit purchase cost
     * @param h    holding cost
     * @param l    lead time from DC
     * @param s    stockout cost per unit
     * @param z    safety stock parameter
     *
     */
    public RetailForecaster(double o, double u, double h, int l, double s, double z, double
a) {
        ORDER_COST = o;
        UNIT_PURCHASE_COST = u;
        HOLDING_COST = h;
    }
}
```

```

    LEAD_TIME_FROMDC = 1;
    STOCKOUT_COST_PERUNIT = s;
    SAFETY_STOCK_PARAMETER = z;
    ALPHA = a;
}

/**
 * @param uiStar forecasted weekly customer demand
 * @param di      actual weekly customer demand
 */

public double computeOtstar( Vector uiStar, Vector di){
    int sum = 0;
    for(int i = 0; i < di.size() ; i++)
    {
        int q = ((Integer)uiStar.elementAt(i)).intValue()-
((Transaction)di.elementAt(i)).qty;
        sum = sum+ Math.abs(q);
    }
    Otstar = sum/di.size();

    System.out.println("Otstar from forecaster: " + Otstar);
    return(Otstar);
}

/** Compute Dstar1
 * @param dstar1 forecasted customer demand in week t+1
 * @param dt      real customer demand in week t
 * @param dtstar  forecasted customer demand in week t. This value is the
 *                same value as dt in week 1.
 */
public int computeDstar1(int dt, int dtstar) {
    dstar1 = (int)Math.round(ALPHA*dt + (1-ALPHA)*dtstar);
    System.out.println("forecasted demand from forecaster:"+ dstar1);
    return dstar1;
}

/** Compute et
 */
public int computeEt() {

    et = (int) (dsstar1*(LEAD_TIME_FROMDC) +
SAFETY_STOCK_PARAMETER* Math.sqrt(LEAD_TIME_FROMDC)*Otstar);
    System.out.println("et-maximum inventory target from forecaster:"+et);
    System.out.println("Otstar from forecaster: " + Otstar);
    System.out.println("forecasted demand from forecaster:"+ dstar1);
}

```

```

        return et;
    }

    /** Compute qt (replenishment order).
     * @param o on-hand inventory level
     * @param b back order from DC
     */
    public int computeQt(int o, int b) {
        qt = 0;
        if (et >= (o+b)){
            qt = et-(o+b);
        }
        System.out.println("retailer replenishment order to DC from forecaster:"+ qt);
        return qt;
    }
}

```

## Supplier.java:

```
import java.util.*;

public class Supplier {

    /** Creates a new instance of Manufacturer */
    Vector mfOrders;
    Transaction submittedOrder;
    Manufacturer manufacturer;
    public Supplier() {
        mfOrders = new Vector();
        manufacturer = null;
    }

    public void setManufacturer(Manufacturer m) { manufacturer = m; }

    public void takeOrder(Transaction o) {
        submittedOrder = new Transaction(o.week, o.qty);
        mfOrders.add(submittedOrder);
    }

    // process order received from Manufacturer
    public void processOrders(int w) {
        // update distributor's buffer
        manufacturer.insertSPSupplyIntoBuffer(submittedOrder.qty);
        System.out.println("SP, submittedOrder.qty:"+submittedOrder.qty+
            ", stockLevel.qty:"+submittedOrder.qty);
    }
}
```

## Transaction.java:

```
public class Transaction {  
  
    int week;  
    int qty;  
    double qy;  
  
    public Transaction(int w, int q) {  
        week = w;  
        qty = q;  
    }  
  
    public Transaction(int w, double p) {  
        week = w;  
        qy = p;  
    }  
  
    public String toString() {  
        return "week:"+week+", qty:"+qty;  
    }  
}
```