AN EVENT-TRIGGER-RULE BASED SUPPLY-CHAIN MANAGEMENT SYSTEM OVER THE INTERNET

By

RAKESH LODHA

A THESIS PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2002
Copyright 2002

By

Rakesh Lodha
Dedicated to
my dearest mother and father
ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Stanley Y.W. Su, chairman of my supervisory committee, for his continuous guidance and support throughout my studies. I am very grateful to him for providing me with a challenging topic for my thesis and for his continuous feedback and support during the implementation and thesis writing. I also would like to thank Dr. Herman Lam and Dr. Sherman Bai for serving on my supervisory committee.

I would like to thank our secretary Sharon Grant for maintaining an excellent work environment in the Database Systems Research and Development Center and for her help in correcting my English in the thesis. I thank Heifei Li for his help during implementation of my thesis. I also thank Dr. Max Shen, Gang Chen, Laith Abu Hilal and Muthukrishnan Shrinivasan for their contribution in the design of the supply-chain management system scenarios.

On a more personal note, I want to thank my family without their love and support, none of this would have been possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ACKNOWLEDGMENTS</th>
<th>iv</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
</tbody>
</table>

## CHAPTER

1. INTRODUCTION .......................................................................................................................... 1

2. ANALYSIS OF CURRENT SCM PRACTICES ............................................................... 9

   Introduction.......................................................................................................................... 9
   Enterprise Resource Planning (ERP) .................................................................................. 10
   Supply-Chain Management ............................................................................................. 11
   Advanced Planning and Scheduling (APS) ....................................................................... 12
   Collaborative Planning, Forecasting and Replenishment (CPFR) .................................. 13
   SCM and Its Advantages ................................................................................................. 14

3. SCM MODEL AND DIFFERENT SCENARIOS USED IN SCM SUPPLY-CHAIN MANAGEMENT SYSTEM .................................................................................................................. 17

   SCM Model .......................................................................................................................... 17
   Different Scenarios In SCM ............................................................................................. 19
      Inventory Replenishment for Retailer ............................................................................. 20
      Quote and Order for Retailer ...................................................................................... 21
      Determine Qualified Vendors ...................................................................................... 24
      Inventory Replenishment for Distributor ..................................................................... 25
      Quote Analysis for Distributor ................................................................................... 26
      Process Order for Distributor .................................................................................... 28
      Quote and Order for Distributor .................................................................................. 29
      Inventory Replenishment for Manufacturer .............................................................. 29
      Automated Negotiation Process ................................................................................. 29
      Dynamic Negotiation Process .................................................................................... 30
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Implemented Events in SCM</td>
<td>34</td>
</tr>
<tr>
<td>2 Example Rules in SCM</td>
<td>36</td>
</tr>
<tr>
<td>3 Example Triggers in SCM</td>
<td>38</td>
</tr>
<tr>
<td>4 Description of Objects in SCM and their Attributes</td>
<td>47</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A simple supply-chain</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>SCM Model</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>Inventory Replenishment for Retailer</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>Quote and Order for Retailer</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>Determining Qualified Distributors</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>Inventory Replenishment for Distributor</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>Quote Analysis</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>Process Order for Distributor</td>
<td>27</td>
</tr>
<tr>
<td>9</td>
<td>Quote and Order for Distributor</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>Inventory replenishment for Manufacturer</td>
<td>29</td>
</tr>
<tr>
<td>11</td>
<td>Two Tier Scenario</td>
<td>31</td>
</tr>
<tr>
<td>12</td>
<td>Description of the SaveQuote Rule</td>
<td>37</td>
</tr>
<tr>
<td>13</td>
<td>SCM Information Infrastructure</td>
<td>40</td>
</tr>
<tr>
<td>14</td>
<td>Sample of a Configuration File for the SCM Database</td>
<td>42</td>
</tr>
</tbody>
</table>
Abstract of Thesis Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Master of Science

AN EVENT-TRIGGER-RULE BASED SUPPLY-CHAIN SYSTEM OVER THE INTERNET

by

Rakesh Lodha

Chairman: Dr. Stanley Y.W. Su
Major Department: Computer and Information Science and Engineering

This thesis presents the design and implementation of an Event-Trigger-Rule based supply-chain management system called SCM. The SCM is constructed by a network of Knowledge Web Servers, each of which consists of a Web server, an Event Manager, an ETR Server, a Knowledge Profile Manager, a Persistent Object Manager, a Metadata Manager, a Negotiation Server, and a Cost-Benefit Evaluation Server. Together, they form a sophisticated supply-chain management system, which manages the activities and interactions among Manufacturers, Distributors and Retailers. The SCM offers a number of advantages over the existing supply-chain management systems. First and foremost is the flexibility offered to business entities in defining their own rules according to their own business strategies. Second, since the rules that control the business activities are installed and processed by the multiple copies of the ETR server installed at business entities’ individual sites, their privacy and security are safeguarded. Third, SCM’s event, event filtering and event notification mechanisms keep both Buyers
and Suppliers better and more timely informed of business events so that they or their software systems can take the proper actions in different business processes.
CHAPTER 1
INTRODUCTION

A supply-chain is made up of all the activities that are required to deliver products to customers from designing the product to receiving orders, procuring materials, marketing, manufacturing, logistics, customer service, shipping, receiving payment and so on. Anyone or anything that influences a product’s time-to-market, price, quality and delivery, among other activities, is relevant to supply-chain management.

In collaborative e-business, a number of business enterprises interact to do a joint business over the Internet. An e-supply chain is a good example of collaborative e-business, which is the focus of this thesis. E-supply chain management is digitally connecting the entire world into one big network of supply-chains. Internet capabilities have already changed and will continue to fundamentally change business-to-business supply-chain models.

For many companies, it has become clear that a supply-chain in which information and material flow with ease can be a significant competitive edge. There is a need to transform from dysfunctional and unsynchronized decision-making – which results in dispersed and very costly supply activities – to a supply-chain that performs in such a synergistic way that it provides the participating company with a competitive advantage.

An effective supply-chain management system is needed to provide the appropriate amount of information to those who need to know it in a timely manner. An
e-supply chain allows customers and suppliers to seamlessly link together, exchanging information almost instantly. The benefits in cycle time compression, lower inventories, decision-making quality and reduced overhead costs make e-supply chain management a highly desirable strategy.

In an integrated supply-chain, customers and suppliers become mutually dependent by collaborating through the sharing of relevant information and by focusing on perfecting the demand and supply process. The objective is for everyone in the supply-chain to increase the market-share through quick response to customers’ needs. This can only happen when information, materials and products flow smoothly, freely, and in synchronization with demands.

The objectives of an improved method of supply-chain management are twofold. Both objectives will affect the cost and revenue sides of the business equation. They are

- To achieve better cost efficiency through a high velocity of information and material flow, thus maintaining lower inventories and thereby eliminating unnecessary overhead activity costs;
- To maintain and preferably increase the market-share of business by being more flexible, efficient and dependable.

The current web and distributed object technologies provide a basic information infrastructure over the Internet to interconnect enterprises and allow data application systems to be shared among customers, suppliers and business partners. However, there are some limitations in the basic information infrastructure when it is used for collaborative e-supply chain management. We shall discuss them in the following paragraphs.

One problem with the existing infrastructure is that it only enables the establishment of a data network instead of a knowledge network. There is very little
knowledge that exists on the Internet or intranet to help in the extraction of meaningful data; timely delivery of data; activation of the correct application systems to process the data; timely notifications on the occurrences of events; automatic execution of rules that enforce data integrity and security constraints, business policies and strategies, government regulations, etc. Here, we define information as data of value used in decision-making, and knowledge as high-level specifications that are useful for making, not just any decisions, but the right decisions. Knowledge captures the successful experiences of enterprises and helps to make business processes automated. In this work, we propose using an enhanced information infrastructure to capture and share the knowledge of business enterprises in supply-chains. In this system, business knowledge is represented by high-level specifications of business events, rules, triggers, and processes that are associated with distributed objects.

The second limitation of the basic infrastructure is that whenever some data is required upstream or downstream of a supply-chain, it has to be pulled. The Internet is based on the client-server paradigm of computing. Servers on the Internet are initially isolated from each other and only respond to requests that are given to them. The pull model of data access requires a system that is doing the pulling to know what to pull and when to pull. Also this pull model requires that the right to access data be given to other entities in a supply-chain, which is highly undesirable as nobody wants to share their business (private) data with other entities. Another problem with the pull model is that it can only work with one-to-one interaction, which is highly inefficient because processing power and bandwidth are often wasted from pulling, especially when performing collaboration among multiple systems. In this work, we propose to use the technology
developed at the Database Systems Research & Development Center, University Of Florida, which combines the features of the push and pull models of information access and delivery and provides more advanced event filtering and composite event processing capabilities. The ETR server is the system that has been implemented using this technology. It provides an expressive modeling language/tool for defining business events, triggers, and rules. By using this tool, business strategies, constraints, and regulations of different entities can be defined in terms of distributed objects having events, triggers and rules. The inclusion of events, triggers, and rules in object class definitions is an extension to the traditional distributed object model and makes distributed object active (i.e., active distributed objects) in that an activation of an operation (or method) of a distributed object can post events to automatically trigger the processing of business rules. These rules are high-level specifications and, hence, are easy for non-technical users to readily understand and make the necessary changes. These rules are not hard-coded in the application systems. This allows the business entities to define their business strategies in terms of these rules. This gives them flexibility in changing their business strategies even after the system is built.

For companies across a supply-chain to coordinate their product, financial and information flows, they must have access to accurate and timely information reflecting the status of their supply-chain. The capability for all supply-chain partners to have access to the shared information on a timely basis is therefore a key to improving supply-chain performance. Information sharing is the most effective way to counter the problem of demand information distortion in a supply-chain -- the well known “bullwhip effect.” Information distortion often arises when partners make use of local information to make
demand forecasts and pass them to upstream partners; partners making ordering decisions based on local economic factors, local constraints or performance measures; and gaming behaviors to exaggerate orders when there are perceived uncertainties in supply conditions. These distortions are amplified from one level to another in a supply-chain, and are considered to be one of the biggest causes of inefficiencies in a supply-chain. One way to counter the bullwhip effect is to have transparency of demand information. A supply-chain management system should guarantee that such information is accessible by the appropriate parties on a real-time basis, which would reduce the possibility of the bullwhip effect. Also while sharing information, secrecy of each partner’s confidential data has to be maintained.

In addition to the server for managing business events, triggers and rules, we believe that additional servers and tools are needed to build a powerful information infrastructure for supporting supply-chain management. For example, when a Buyer receives quotes from Suppliers, these quotes need to be evaluated in order to find the best quote that satisfies the cost and benefit requirements of the Buyer. Thus, a Cost-Benefit Evaluation Server is needed to perform cost-benefit evaluation of the received quotes and a tool for capturing the Buyer’s preference and cost information, with respect to the contents of each quote will also be necessary. Also a Buyer may want to negotiate with a Supplier over the price, the quantity, the delivery date, and the attributes of a product or service. A delivery truck’s accident may require the re-negotiation of a new delivery schedule, which may involve negotiations over the schedules up and down a supply-chain. Also these negotiations are not just one-on-one negotiation. They may involve negotiations with multiple parties simultaneously (i.e., multilateral negotiations). It
would be beneficial if a part or all of the negotiation tasks could be carried out automatically over the Internet as a part of e-business activities. Thus, the services of a Negotiation Server are needed to perform automated negotiations. A tool for capturing the product/service constraints of both Buyers and Suppliers is also needed for the analyzing negotiation proposals and the generation of counter-proposals.

In a supply-chain management system, these servers may depend on each other for services. For example, when a Buyer receives multiple quotes from Suppliers (an event); the ETR server is notified of the event, which triggers the Cost-Benefit Evaluation Server to evaluate the quotes to do Supplier selection. The Buyer may want to negotiate with a selected Supplier for the term and condition of the purchase. In that case, a Negotiation Server installed at the Buyer site will be activated to perform automated negotiation with the Negotiation Server installed at the Supplier’s site. During the negotiation process, if the conditions specified in a negotiation proposal do not satisfy the constraints of the Buyer/Supplier, a rule that implements a negotiation strategy and is managed by the ETR Server, may be triggered to do some concession before a counter-proposal is generated and sent to the Supplier/Buyer. Proposals and counter-proposals can be sent back and forth between the two Negotiation Servers until either an agreement is reached by both negotiation parties or one party unilaterally terminates the negotiation process. The acceptance and the termination of a negotiation can again be determined by rules.

In a supply-chain management system, there are many situations in which one business process depends on another business process’ result. Thus that business process has to wait until the other process is finished; this needs to be decided at runtime.
Therefore a system is required which can determine these needs and adjust itself accordingly at runtime. In this work we show that our system can adapt to the system’s runtime behavior through the use of rules. In this work, we show that one negotiation process can determine whether it needs to wait for another negotiation process to finish so that it can use the other negotiation process’ result as part of its negotiation.

In this work, we integrate an ETR Server [LEE00, LEE01,SU00b], a Cost-Benefit Evaluation Server [SU87, YU01], a Negotiation Server [HAM00, HUA00, LI01,SU00a, SU01], a Timer Event Server, a Constraint Satisfaction Processing (CSP) Server [LI01] and their associated GUI tools which have been developed at the Database Systems Research and Development Center to form an information infrastructure for supply-chain management; and we develop a number of supply-chain scenarios for use to demonstrate how these servers can be used to support supply-chain management. The developed scenario involves three business entities: a Retailer, a Distributor and a Manufacturer. By using these three entities, we are able to test and demonstrate the interactions of one business entity with two adjacent neighbors. The same business activities and interactions can occur in any segment of a supply-chain. Thus, the results of this work can be applied to a supply-chain with more than three business entities.

The contributions of this work are 1) the integration of several existing servers and tools to construct a supply-chain management system (SCM), 2) the development and implementation of several supply-chain scenarios by using the services of the integrated servers and tools, 3) the testing and demonstration of the developed SCM system.
This thesis is organized as follows. In Chapter 2, we review the basics of supply-chains and their changing needs due to recent changes in business models, the development of the Internet, and the rise of new software applications. We analyze the current practices in supply-chain management – the evolution of enterprise applications (Enterprise Resource Planning) to inter-enterprise applications (Advanced Planning and Scheduling, Collaborative Planning, Forecasting and Replenishment). We also motivate our development of a supply-chain management system SCM. In Chapter 3, we explain the SCM model and the scenarios we have developed to demonstrate the utilities of the servers in the proposed information infrastructure. In Chapter 4, we describe the architecture of SCM and different technologies used in its implementation. Chapter 5 provides some implementation details. Finally, in Chapter 6, we conclude by pointing out the benefits of the integrated system and propose issues for further research.
CHAPTER 2
ANALYSIS OF CURRENT SCM PRACTICES

2.1 Introduction

The past three decades have seen tremendous developments in software for production and operations management. This has been a direct result of the advancement in computers and information technologies, and also of the new way companies run their businesses. There has been a distinct trend of increased collaboration among business organizations and their trading partners over the years.

By late 1980s, companies found an increasing need to integrate information available in different units of an organization in order to make better decisions, improve productivity and increase profits. This led to the development of enterprise resource planning (ERP) applications. With competition increasing with time, companies soon found it necessary to optimize the entire product “supply-chain”. This called for collaboration, not only within the organization, but also with trading partners in the supply-chain. The importance of managing customer relationships, being flexible to respond to changes in organizational structure as well as customer demand, managing the product life cycle, etc., influenced the growth of next generation software applications - advanced planning and scheduling (APS). This software used optimization algorithms to compute the optimal production plan and machine schedules to reduce operating costs and to improve profits. Now companies are looking for ways to reduce lost sales, to match supply and demand with lower inventory, while remaining as lean as possible.
They are also adapting a new concept called collaborative forecasting, planning and replenishment (CFPR) to achieve the above.

In this chapter, we review the developments from ERP to CPFR. We analyze the motivation, structure, examples, benefits, and drawbacks for each of these applications and explain how our system when integrated with these systems helps to implement an efficient supply-chain management system.

2.2 Enterprise Resource Planning (ERP)

An ERP system is a configurable information system package that integrates information and information-based processes within and across functional areas and multiple sites in an organization. An ERP package has many detailed options, parameters, capabilities and models built into it when developed by a vendor. An organization implementing an ERP package must clearly understand, identify, and outline its objectives in this implementation, and its requirements and expectations from the ERP package.

By implementing ERP applications, organizations aim to replace complex, disparate, obsolescent systems, improve competitive performance, and improve the quality and visibility of information. ERP applications help organizations track customers, money, materials, assets, labor, utilization, etc.

ERP systems have their shortcomings. They are built for recording events that have already occurred, rather than planning for the future. Thus, they are good at record keeping but not at intelligent decision-making. They can accommodate complex workflows, but lack the ability to adapt and restructure with changes in surroundings. They also lack the ability to scale to large volumes since their order-taking capacities are
limited. While they can integrate multiple business functions, they lack the ability to expand their scope to multiple enterprises.

In the next section, we analyze supply-chains and the changing objectives of a company as a trading partner in today’s supply-chains.

2.3 Supply-Chain Management

The original motive of Supply-Chain Management was “elimination of barriers between trading partners” in order to facilitate synchronization of information between them.

Figure 2.1 A simple supply-chain.

Figure 1 shows the structure of a typical supply-chain. It consists of a number of units beginning with Suppliers, who provide raw materials to factories or Manufacturers, which manufacture products and send them to Distribution Centers. These centers transport the products to regional Distributors or Wholesalers, which in turn, ship the products to Retailers. The end of a traditional supply-chain is usually the customers, who buy products from the Retailers. Although this composition is typical, supply-chains vary in length. Different industries might have slightly different structuring of their supply-chain. Traditionally, planning, purchasing, manufacturing, distribution, and
marketing operated independently along the supply-chain. Each activity had its own set of objectives and often, these objectives were conflicting (for example manufacturing operations may be designed to maximize throughput and minimize costs, with little consideration for inventory levels, distribution capabilities or market demand). Supply-chain management has evolved as a strategy to coordinate the activities of these independent functions, and create a single, integrated plan for the entire organization.

The objective of supply-chain management activities is to meet customer demand for guaranteed delivery of high quality, low cost, customized products with minimal lead-time. The attempt is to improve responsiveness, understand consumer demand, intelligently control the manufacturing process, and align together the objectives of all partners in the supply-chain.

These needs have propelled the development of optimization packages for managing the entire business process, beginning with advanced planning and scheduling (APS) tools to help companies match their supply with demand, and later integration with modules for customer relationship management and product lifecycle management. We begin with a review of APS tools in the next section.

2.4 Advanced Planning and Scheduling (APS)

Supply-chain management requires responsive, intelligent decision support tools to determine optimal (or at least feasible) methods of satisfying customer demand and product supply. APS tools have been developed with the intention of meeting this requirement. These systems aim at optimizing the supply-chain objective discussed in the previous paragraph subject to such constraints as resource availability, capacity costs, labor and materials costs, and transportation resources. They help companies forecast
demand with the help of sophisticated modeling and statistical techniques. APS tools are designed to help companies create plans and schedules that are based on system constraints. APS tools generate a high rate of return by shortening the forecast cycles, enhancing visibility of production plans and schedules, increasing accuracy of order date commitments, taking real-time decisions in the face of supply/demand fluctuations, and planning in real-time rather than batch processing.

Although APS tools are “intelligent” they are hampered due to their focus on the manufacturing, distribution and transportation functions in the supply-chain. This focus is acceptable in traditional, slow-moving environments. The current fast-paced business environments warrant a broader scope of planning efforts. Where customization and perfect delivery are the price for getting business, customer interaction is the main driver behind the entire product delivery process. Software vendors have moved on from pure APS tools to building a complete set of software for business process optimization.

2.5 Collaborative Planning, Forecasting and Replenishment (CPFR)

We observe an increasing trend towards improving demand forecasting and fulfillment through the development of CPFR. CPFR attempts to determine the right number of specific products to put in an individual store on a particular day of the year. Unpredictable factors such as weather, transportation delays, production problems, and administrative errors can all wreck havoc on supply and demand. Product promotions create massive swings in demand. Suppliers are forced to carry unprecedented amounts of safety stock, or to stay lean and risk being unable to fulfill demand. The first option raises costs for everyone; the second results in lost sales, and frustrates customers. Also, reducing inventory costs returns hidden savings in the supply-chain, due to factors such
as freeing up of fixed assets, capital costs for inventory, manufacturing inefficiencies, redundant handling and transportation, and improved customer service.

CPFR takes an integrated approach to supply-chain management among a network of trading partners. Trading partners share forecast and results data. CPFR software analyzes this data, and alerts planners at each company to exceptional situations that could affect delivery or sales performance. Trading partners then collaborate to resolve these exceptions by adjusting plans, expediting orders and correcting data entry errors.

If implemented correctly, CPFR improves data communication among trading partners. It improves forecasting and planning by providing the ability to see planned results. A Retailer can prevent out-of-stock situations, especially during product promotions. A Manufacturer can optimize product mix, promotion timing, and margins across supply-chains. Better partnership with Suppliers results in lower inventory levels for Retailers. Manufacturers can make-to-demand rather than make-to-stock, and offer savings in inventory and production costs, and in product obsolescence costs.

2.6 SCM and Its Advantages

Our system helps companies to achieve greater coordination and collaboration among supply-chain partners via “supply-chain integration”. In our system, the Internet plays a key role in furthering the goals of supply-chain integration. With the help of the Internet, companies can redefine how back-end operations — product design and development, procurement, production, inventory, distribution, after-sales service support, and even marketing — are conducted; and, in the process alter the roles and relationships of various parties, fostering new supply networks, services, and business
models. Our system helps companies to integrate their supply-chain with which they can realize dramatic returns through efficiency improvements, better asset utilization, faster time to market, reduction in total order fulfillment times, enhanced customer service and responsiveness, penetrating new markets, and ultimately higher return on assets.

Our system uses an Active Object Model (AOM) and other technologies such as Cost-Benefit Evaluation, and automated Negotiation in implementing SCM. With these technologies the interaction between different entities in a supply-chain is implemented using events, triggers, and rules. This allows these entities to have real time visibility of the system, making each business process efficient and, hence, the whole supply-chain. In later chapters we will explain in detail how our system uses these technologies to make supply-chain efficient.

There are two key dimensions in which the impact of our system can be found:

- Information integration
- Planning synchronization

Information integration refers to the sharing of information among members of a supply-chain. This includes any type of data that could influence the actions and performance of other members of the supply-chain. Some examples include: demand data, inventory status, capacity plans, production schedules, promotion plans, and shipment schedules. Information integration is the foundation of supply-chain integration. Our system makes sure that such information is accessible by the appropriate parties on a real-time basis via events, triggers, and rules, and, hence, reduces the possibility of bullwhip effect.
Planning synchronization refers to the joint design and execution of plans for product introduction, forecasting, and replenishment. In essence, planning synchronization defines what is to be done with the information that is shared; it is the mutual agreement among members as to specific actions based on that information. Hence, members in a supply-chain may have their order fulfillment plans coordinated so that all replenishments are made to meet the same objective – the ultimate ideal response to customer demands.

Our system helps companies to achieve these key dimensions provided the implementation of the supply-chain is flawless. The success of any supply-chain integration effort is predicated on close cooperation and inspired by a perception of mutual benefit.
CHAPTER 3
SCM MODEL AND DIFFERENT SCENARIOS USED IN THE SCM SUPPLY-CHAIN MANAGEMENT SYSTEM

This chapter describes an SCM model and different scenarios developed for the SCM. The SCM model is used for the specification and implementation of our web-based supply-chain management system.

3.1 SCM Model

In order to demonstrate the use of SCM infrastructure technologies for an e-supply-chain management system, we developed a generic model; and with the help of this model, different generic scenarios have been developed for experimental
implementations. These scenarios will be explained in detail in the following sections. The SCM model consists of different entities consisting of many Buyers and Suppliers and it shows how business is accomplished among them. There can be many Buyers interacting with the same Suppliers. Figure 2 shows this abstract model irrespective of the technology used to implement it.

According to this model, a Buyer sends a Request-for-Quote (RFQ) to Supplier. This RFQ may be sent to one or many Suppliers. It contains the requirements of the Buyer such as product name, product number, quantity of product, delivery date, etc. Upon receiving the RFQ, each supplier checks its business policy. The Supplier also checks its inventory and, if inventory is not enough, then it in turn sends an RFQ to its Suppliers (i.e. Supplier’s Suppliers). A Supplier’s Supplier sends its quote to the Supplier and, based on this quote’s information, the Supplier sends its quote to the Buyer. The Buyer then checks different quotes received from different Suppliers and does a cost-benefit evaluation on each of them. After this cost-benefit evaluation, if no quote matches with the Buyers expectations, then it can start a negotiation with different Suppliers simultaneously. Also the Supplier can check whether he has sent request for quote to its suppliers and based on those quotes it can start negotiation with its suppliers. Once that negotiation is finished successfully, the Supplier can use its results to continue the negotiation with the Buyer. Buyer/Supplier can define its own business rules used in the negotiation. If the negotiation is unsuccessful, then the Buyer can resend a modified RFQ to the Suppliers and the whole process can start over again. Upon a successful negotiation, the Buyer sends a purchase order to a respective Supplier. The Supplier in
turn sends a purchase order to its respective Supplier. Upon receiving the shipment from the Supplier’s Supplier, the Supplier sends the product to the Buyer.

The above model represents two links (three business entities) of a long supply-chain. It shows that the business interactions between two entities in one link can affect the interactions between two entities in another link (e.g., the result of a negotiation in one link can affect the negotiation of the other link.) The business interactions shown in the model can similarly occur in other segments of a supply-chain.

3.2 Different Scenarios in the SCM

To verify the effectiveness of the SCM model for e-supply chain applications, we have developed scenarios for experimental implementations. The purpose of these scenarios is the following.

- To demonstrate a prototype of the integration of inter-enterprise business functions within the scope of the SCM model.

- To demonstrate the integration of several infrastructure components in support of a supply-chain management system.

- To demonstrate how the different technologies such as an Event-Trigger-Rule technology, an automated negotiation technology, a cost-benefit evaluation technology, and a constraint satisfaction processing technology can be integrated in support of a supply-chain management system.

These scenarios include major players of supply-chains, such as Retailers, Distributors, and Manufacturers, and describe business processes which include determining qualified Suppliers, requesting quotes, and automating negotiations between any two entities in the supply-chain, as well as inventory replenishment, quote evaluation, transporting decision, cost-benefit evaluation, and order processing for any entity in the chain. These scenarios do not show what specific software or technology is used to implement them. This allows different supply-chain entities to use different
tools, systems and methodologies to implement the scenario and demonstrate plug-and-play and different approaches to solve the supply-chain problems.

There is a trend for supply-chain entities to explicitly specify conditions or areas of interest and business policies, constraints, regulations, strategies, etc., in terms of business events and business rules. These events and rules become important enterprise resources that need to be shared across collaborative business enterprises. In a supply-chain, each business entity has its own business strategies and policies. Our scenarios allow each business entity to define its own strategies and policies in the form of events and rules giving it more flexibility in sharing its data with other entities in the supply-chain. We will explain each scenario in detail in the following sections.

3.2.1 Inventory Replenishment for Retailer.

This business process identifies some of the initial activities that are involved during a procurement cycle. The Buyer’s procurement system needs to determine the order policy for a part and takes the appropriate action when a replenishment trigger is received by it. The UML Activity Diagram for Inventory Replenishment process steps is shown in Figure 3.

- **Analyze inventory with respect to requirements.** For a product that is supplied by Suppliers, a routine analysis of inventory availability with respect to part requirements is generally performed by the Buyer's system. The procurement process for such a part is triggered when the Buyer's system creates a purchase requisition for the part.

- **Initiate Quote and Order.** If an outstanding purchase order does not exist for a part then the quote and order process for a new purchase order has to be initiated. This process is described in more detail in the following "Quote and Order" section.

In our implementation of this scenario, the ETR Server is used to post the RFQ (Request-for-Quote) event to the Supplier.
Figure 3 Inventory Replenishment for Retailer

3.2.2 Quote and Order for Retailer

This business process applies to the procurement of all types of products. The purchase order for a product can be valid for a one time purchase or it can be a blanket purchase order covering multiple deliveries over a period of time. The UML Activity Diagram for the Quote and Order Entry process steps is shown in Figure 4

- **Determine Qualified Distributors.** Potential Distributors for a product have to be found before the Retailer can request price quotations. The process to find suitable Distributors is described in more detail in the section on "Determine Qualified Distributors".

- **Request For Quote.** After finding suitable potential Distributors for a product, Request-For-Quote (RFQ) is sent to the potential Distributors of the product. All Distributors selected for a specific product are given the relevant data of the
product in the RFQ. The relevant data may include engineering drawing (in a standard non-CAD format such as GIF/TIFF/etc.), products list, process specifications, schedules, and quantity requirement.

- **Check Inventory.** The Distributor procurement system needs to determine the inventory on hand of a product before sending any quote to a Retailer. This process will be explained in detail in the section on “Inventory Replenishment for Distributor”. The “Inventory replenishment algorithm” is called after a Distributor receives the RFQ from the Retailer.

![Process Flow Diagram](image)

**Figure 4. Quote and Order for Retailer**

- **Send Quote.** The Distributors send their responses to requests for quote ("Reply RFQ"). If enough capacity and materials cannot be made available by a Distributor to satisfy the customer's delivery requirement then the Distributor can respond with a delivery schedule that is different from the original RFQ. This may lead to a negotiation between the Distributor and the Retailer.
In our implementation, this response is sent to the Buyer with the help of the ETR Server by posting a “Reply RFQ” event.

- **Evaluate Quote.** After evaluating responses to RFQ ("Reply RFQ") from all Distributors, the Retailer reviews the quotes to select the winning Distributor.

- **Negotiate.** If no quote matches the Retailer’s constraints and requirements, then the Retailer starts a negotiation with the Distributor that offers the best quote (i.e., the best cost-benefit value). This process is discussed in detail in the following sections.

In our implementation, a pair of replicated Negotiation Servers are used to do the negotiation between the Retailer and the Distributor.

![Diagram of determining qualified distributors](image)

**Figure 5 Determining Qualified Distributors**

- **Generate Purchase Order.** After receiving the responses to the RFQ (Reply RFQ business objects) from all Distributors, the Retailer reviews the quotes and selects the best quote and hence the Distributor with which it wants to do business. A purchase order for that product is generated and stored in the Retailer’s database.
for planning purposes. The Retailer then authorizes its receiving department to accept the shipment against the purchase order. This purchase order is then sent to the winning Distributor.

- **Order Entry.** The Distributor saves the Retailer’s purchase order in its system.

- **Process Order.** The Distributor posts this purchase order to his sales department. Once that order is processed, the inventory of the Distributor reduces which in turn triggers the inventory control system of the Distributor.

- **Initiate Shipping.** The shipping process is initiated here.

- **Acknowledge Purchase Order.** The Distributor acknowledges the receipt and acceptance of the purchase order.

- **Process Acknowledge PO.** After receiving the PO acknowledgement the Retailer reviews the delivery schedule to ensure that it meets the ordered product requirements.

### 3.2.3 Determine Qualified Vendors

The UML Activity Diagram for "Determine Qualified Vendors" process steps is shown in Figure 5.

- **Obtain a list of qualified Distributors.** The Retailer obtains a list of qualified Distributors that have previously supplied various products.

- **Qualified Distributor Found** For a product, existing qualified vendors for the product are found from the list of qualified Distributors based on some selection rules for the applicable project or program.

  For a new product, existing qualified Distributors cannot be found from a simple list of Distributors. It may be possible to find existing qualified Distributors that have the capability to supply the new product by using a Supplier capability matching process.

- **Use External Services to Find New Suppliers.** If no Distributor meets the product requirements new Distributors can be found by human intervention, the Yellow Pages, Internet advertisements etc.
3.2.4 Inventory Replenishment for Distributor

The Distributor’s inventory Replenishment processes are the same as that of the Retailer. Here the Distributor acts as a Buyer and the Manufacturer acts as a Supplier. The diagram for the inventory replenishment process steps is shown in Figure 6.

In our implementation, if there are not enough inventories, then an RFQ is posted to the Manufacturer via the RFQ event using the ETR Server.

![Diagram of Inventory Replenishment Process]

Figure 6. Inventory Replenishment for Distributor

3.2.5 Quote Analysis For Distributor

After receiving all the quotes from the Distributors, the Retailer needs to analyze each quote and decide the best quote so that a purchase order can be sent to the respective Distributor. The diagram for the quote analysis process steps is shown in Figure 7.
- **Get the quote.** Get all the quotes from the Retailer database which stores all the quotes from Distributors.

- **Compare against constraints.** Compare these quotes with the constraints defined by the Retailer. These constraints define different business requirements of the Retailer.

- **Do a cost-benefit evaluation.** This process determines the cost and benefit of each quote received by the Retailer. Each quote is then given an aggregated value.

  In our implementation, the cost-benefit evaluation is done using the Cost-Benefit Evaluation Server.

![Quote Analysis Diagram](image)

**Figure 7. Quote Analysis**

- **Sort quotes by cost-benefit values.** This process sorts all the quotes by the aggregated values assigned to these quotes in the cost-benefit evaluation. Then these quotes are stored in the Retailer database in the sorted order.
3.2.6 Process Order for Distributor

The UML Activity diagram for the "Process Order for Distributor" process steps is shown in Figure 8. Inventory on the Distributor’s side for that particular product is reduced and then inventory control module is called, which checks the inventory against the inventory policy. If inventory replenishment is required, then the Distributor starts the “Initiate and Quote process.”

![Diagram of Process Order for Distributor](image)

Figure 8. Process Order for Distributor

3.2.7 Quote and Order for Distributor

This business process is similar to “Quote and Order for Retailer”. Here the Distributor acts as a Buyer and the Manufacturer acts as a Supplier. The UML Activity Diagram for the Quote and Order for Distributor process steps is shown in Figure 9. The
only difference here is that on the Manufacturer’s side, an inventory control system sends the product request to the Manufacturer’s machine shop for producing the product if there are not enough inventories.

Figure 9. Quote and Order for Distributor

3.2.8 Inventory Replenishment for Manufacturer

The Inventory Replenishment processes for the Manufacturer are the same as that of the Retailer except for the “Call scheduling algorithm”. Here, if inventory replenishment is required on the Manufacturer’s side, then the inventory module will call the scheduling algorithm process and send the respective data about product and quantity to this algorithm. This algorithm then decides the production schedule and the quantity
to be produced. The diagram for the Inventory Replenishment for Manufacturer process steps is shown in Figure 10.

Figure 10. Inventory replenishment for Manufacturer

3.3 Automated Negotiation Process

Automated negotiation makes use of the Internet to conduct a negotiation process. Detailed discussion about the negotiation decision process, negotiation phases, and their functions can be found in [LI01].

In order for Buyers and Sellers to make use of the automated negotiation service, they must each register with a negotiation server of their choice and provide the knowledge that is necessary to conduct an automated negotiation. After a Buyer decides
to negotiate with a particular Seller, the Buyer starts the process by submitting an initial product or service request to its negotiation server. The negotiation server A (NSA) initiates a bargaining process by creating and sending a Call-for-Proposal (CFP) to Negotiation Server B (NSB), which represents the Seller. NSB checks the information and constraints specified in the CFP against the registered information of the seller to determine if there are conflicts in requirements and constraints. The conflicting data conditions and constraints can be modified either by using the registered information and knowledge of NSB or through human intervention to produce a proposal to NSA. NSA will then evaluate this proposal against the registered information. In the event that a conflict or constraint violation is found, relevant rules, which have been provided by NSA’s clients, are applied to either (1) reject the proposal and suggest possible changes to NSB, (2) call for human intervention, or (3) generate a counterproposal to be returned to NSB. In the event that the proposal contains a number of alternative data conditions, a cost-benefit evaluation component (i.e., the Cost-Benefit Evaluation Server) is called to pick the best alternative. This alternative is then sent back to NSB, together with NSA’s constraints, as the counterproposal, which starts another round of negotiation. The counterproposal is evaluated in the same way by NSB against its client’s registered requirements and constraints. This process will continue until either an agreement is reached or either side terminates the negotiation process.

3.4 Dynamic Negotiation Process

In e-supply chain infrastructure one-to-one negotiation is not efficient, as it does not give optimum results in a dynamic market-place. To make an e-supply chain fast and efficient, there is a need to have multilateral automated negotiation.
Multilateral negotiation means the ability to do simultaneous negotiations with multiple business entities. Also these negotiations need to be dynamic. Here the word “dynamic" means the ability to take the proper action or to make the proper decision in one negotiation, depending upon the status of other simultaneous negotiations.

![Diagram of Two Tier Scenario](image)

**Figure 11. Two Tier Scenario**

In the TWO-TIER scenario shown in Figure 11, the Retailer first checks the quote given by the Distributor and upon finding some discrepancy it starts a negotiation with the Distributor. Upon receiving a proposal from the Retailer, the Distributor checks the proposal with its policies and constraints and realizes that this negotiation in turn depends on the negotiation with the Manufacturer. So the Distributor has to temporarily suspend this negotiation and first complete the negotiation with the Manufacturer.
There can be many different scenarios in a supply-chain where there is need of dynamic negotiation service. For example, while in the middle of a product delivery if the transportation truck has an accident then the delivery will not be done on the specified date/time; hence a new delivery date/time is to be negotiated. If the Manufacturer’s machinery stops because of a mechanical problem then the Manufacturer will have to negotiate with the Distributor on a new delivery date, which may in turn invoke some simultaneous renegotiations with Retailers.
This chapter describes a knowledge model and the architecture of an event-trigger-rule based supply-chain management system named SCM. The knowledge model is used for the specification and implementation of the Web-based Knowledge Network [LEE01], which provides the information infrastructure and services needed to implement SCM.

4.1 Knowledge Model

In collaborative e-business, it is necessary to have a uniform way to model many types of inter-enterprise resources. In this work, we use an active distributed object (ADO) technology developed at the Database Systems Research and Development Center, University of Florida, which incorporates business events, rules, and processes in the modeling and processing of various types of enterprise resources and provides GUI tools for capturing the active properties of Active Distributed Objects.

The model used in the development of the Internet-based Knowledge Network is an active object model (AOM). In this model, all things of interest are modeled as active objects. In addition to the traditional way of defining an object class in terms of attributes (or properties) and methods, AOM allows the inclusion of events, triggers, and rules in an object class definition. Just as events, triggers, and action-rules can transform a conventional database system into an active database system, they are used in AOM to transform distributed objects (DOs) into active distributed objects (ADOs).
methods are the “incoming” APIs of an object (i.e., they define the functions that can be performed by the object), and events are “outgoing” APIs (i.e., they define the signals that can be generated by the object).

### 4.1.1 Events

Generally speaking, events are things of interest that have happened on the Internet. For example, new data is made available on a site, a web page has been modified, an application system is about to be invoked, a user strikes a key on the keyboard, a signal is received from an external device, etc. Users or software systems can subscribe to events and be notified when events occur. Events can have parameters (i.e., parameterized events), which are data associated with the events that are to be transmitted in event notifications. In the context of a supply-chain, events can be used to represent things that happen during different business processes. For example, a Supplier would like to be notified when a Buyer posts a Request-for-Quote event. This Supplier is registered with the Buyer’s site so that it can receive the Request for Quote for a particular product.

**Table 1. Implemented Events in SCM**

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Parameters</th>
<th>Parameter Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFQ</td>
<td>Quote</td>
<td>String</td>
<td>Event posted when a request for quote is sent by Buyer.</td>
</tr>
<tr>
<td>ReplyRFQ</td>
<td>Quote</td>
<td>String</td>
<td>Event posted when a quote is submitted by supplier</td>
</tr>
<tr>
<td>Timeevent</td>
<td>A</td>
<td>Integer</td>
<td>Event posted when a response for RFQ time limit has expired.</td>
</tr>
</tbody>
</table>
The Supplier can specify different products of interest by providing values to some attributes, which are provided in an event filter template. Similarly, the Buyer would be interested in being notified when the Supplier posts a Send-Quote event. Another point of interest would be when some time has elapsed after sending the Request for Quote. When this point of interest occurs the Buyer may specify some actions to be taken at this point depending on its business strategy.

Table 1 lists the events implemented in SCM, their parameters, and the description of these events.

4.1.2 Rules

In the knowledge model, rules are Condition-Action-ALTaction (CAA) rules, which can be used to define business constraints, policies, regulations, and integrity and security constraints. Each CAA rule represents a small granule of control and logic. A number of related rules can form a rule structure to express a larger granule of control and logic for modeling a more complex policy, regulation, etc. A rule can participate in multiple rule structures, thus making each rule reusable. A rule has a rule name and can have parameters. When the rule is invoked upon the occurrence of some event, it evaluates the CONDITION clause of the rule. If the result is true, the operations specified in the ACTION clause are executed. Otherwise, the operations specified in the ALTACTION clause are executed.

A rule also has a RULEVAR clause, which allows variables to be defined in a rule. The variables can be persistent or temporary. It is also possible to define customizable rule variables, which can be assigned different values for different users, making the rule a “parameterized rule.”
Different from the Event-Condition-Action rules used in some active database management systems [CHA94, DAY98, STO88, SU97 and WD96], events and rules in the knowledge network are separately defined by users or business organizations; and events are tied to rules by trigger specifications (to be described next). This separation is important in a distributed environment in which software systems are loosely coupled. In a supply-chain management system, different entities define and post events, which are subscribed by other entities; these entities may define different rules for implementing different business strategies. For example, every time a Supplier receives an RFQ (Request-for-Quote) event, a rule may be triggered to check the quantity of the requested quote; and if the quantity is small, then notify the Buyer that the Supplier is not interested in doing business because the quantity is too small. These rules are managed and used by an Event server and an ETR server installed at the subscribers’ sites; thus subscribers’ privacy will not be compromised.

Table 2 shows two examples of rules implemented in SCM.

### Table 2. Example Rules in SCM

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Side</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaveQuote</td>
<td>Buyer</td>
<td>Used to save the quote received from Supplier side</td>
</tr>
<tr>
<td>TimerRule</td>
<td>Buyer</td>
<td>Used to stop receiving Quotes after some time is elapsed</td>
</tr>
</tbody>
</table>

A Buyer side rule that defines the Buyer’s business strategy is shown in Figure 12. It should be pointed out that more complex rules, that define different business strategy, can be likewise defined.
RULE

RETURNS

DESCRIPTION

RULEVAR

CONDITION

ACTION

ALT ACTION

<table>
<thead>
<tr>
<th>RULE</th>
<th>SaveQuote (String Quote)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETURNS</td>
<td>void</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>&quot;Checks if buyer has received enough quotes or if time has been passed to receive quote and then saves this quote to db&quot;</td>
</tr>
<tr>
<td>RULEVAR</td>
<td>Int Flag, Int Counter</td>
</tr>
<tr>
<td>CONDITION</td>
<td>[Flag == 1] Counter == 5</td>
</tr>
<tr>
<td>ACTION</td>
<td>RuleLib.UpdateFlag.update(0); RuleLib.RemoveTimerEntry.remove(); RuleLib.EvaluateQuote.evaluate()</td>
</tr>
<tr>
<td>ALT ACTION</td>
<td>RuleLib.UpdateCounter.update(++Counter); RuleLib.InsertQuote.insert(Quote)</td>
</tr>
</tbody>
</table>

Figure 12. Description of the SaveQuote Rule

The SaveQuote rule’s main purpose is to save the quote if it satisfies the condition of the time limit to submit the quote.

The rule has one input parameter – Quote, which is an XML BOD. This parameter corresponds to the parameters of the event “ReplyRFQ”.

As the rule does not return any value, the RETURNS clause of the rule is set to void. The description of the rule is a statement describing the function of the rule.

Rule variables Flag and Counter have been defined in the RULEVAR clause. Flag’s data type is int. Its purpose is to check whether the time limit to submit quote has been passed or not. Its value is stored in a persistent object manager (POM) so that it can be checked later. Also, Counter is used to check whether the Buyer has received the predefined number of quotes. Its value is also stored in POM.

The CONDITION clause is used to evaluate whether the Buyer has received a predefined number of quotes. Here the CONDITION clause is a guarded expression and it is guarded by the expression Flag == 1. If the Flag is not equal to 1, the entire rule is
skipped. If the Flag == 1 and the Counter == 5, then the Action part is processed. If the Flag == 1 and the Counter is not equal to 5, then the Altaction part is processed.

The ACTION section is executed when the value of the Counter is greater than or equal to that of the predefined number of quotes. The TimerEvent entry is removed from TimeServer. Also the Flag’s value is changed to 0 so that no new quotes will be accepted. Then the evaluation of Quotes is done.

The ALTACTION section is executed when the value of the Counter is not equal to the predefined value. In this case the value of the Counter is incremented and then the received quote is stored in POM so that it can be used later to evaluate against other quotes.

4.1.3 Triggers

A trigger relates a structure of events with a structure of rules. An event structure has two parts: namely, a TRIGGEREVENT and an EVENTHISTORY. The TRIGGEREVENT part specifies a number of events. The occurrence (or posting) of any one of these events would initiate the evaluation of the EVENTHISTORY part. The EVENTHISTORY part can be a complex event expression, which makes reference to some events that have already occurred.

Table 3. Example Triggers in SCM

<table>
<thead>
<tr>
<th>Trigger Name</th>
<th>Event</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFQ-RFQRule</td>
<td>RFQ (Posted at Buyer’s side)</td>
<td>RFQRule (Supplier side Rule)</td>
</tr>
<tr>
<td>ReplyRFQ-SaveQuote</td>
<td>ReplyRFQ (Posted at Supplier’s side)</td>
<td>SaveQuote (Buyer side Rule)</td>
</tr>
</tbody>
</table>
“E1 follows E2 follows E8,” and “E4 and E7 occurred within a certain time window” are examples of event history expressions. Thus, upon the occurrence of an event specified in a TRIGGEREVENT expression, the EVENTHISTORY expression is evaluated. If the result is true, then the rule structure given in the trigger specification is processed. Table 3 shows two example triggers.

4.2 Architecture Of SCM

The architecture of SCM is shown in Figure 13. As shown in Figure 13(b), the SCM Node consists of a Knowledge Web Server, a Negotiation Server and a Cost-Benefit Evaluation Server. The Knowledge Web Server consists of a Web server and a number of additional components. These components are: an Event Server, an Event-Trigger-Rule (or ETR) Server, a Knowledge Profile Manager, a Persistent Object Manager [SHE01], and a Metadata Manager. The SCM infrastructure is a network of SCM Nodes deployed at the sites of participating enterprises, much the same way as Web servers are deployed to provide Web services. These servers are replicated and installed at multiple sites. A Knowledge Web Server installed at an SCM site would provide the services needed to maintain all the business process related information.

Figure 13(a). Overall architecture
Knowledge Web Servers installed at the Buyers’ and Suppliers’ sites would provide the services needed for playing the roles as Buyers and Suppliers, respectively. Although we separate Buyers from Suppliers, an organization at a single site can be both a Buyer and a Supplier. The components installed at each site can support the operations and maintain the information for both roles.

### 4.2.1 Event Server

The SCM Node’s Event Server is responsible for accepting event registrations from Buyers/Suppliers and delivering the events (i.e., event notification) to them. The Event Server provides an event registration facility for Buyers/Suppliers to subscribe to the events they are interested in. During the event registration, the Buyers/Suppliers can define their own event filters by selecting or providing values to attributes that are provided in the pre-defined event filter templates.

These filters specify the data conditions under which the Buyers/Suppliers and the Event Servers of their corresponding Knowledge Web Servers would be notified. They allow Buyers and Suppliers to have selective subscriptions to different events.
The Event Server has its APIs (Application Programmers Interface), which allow the programs executing on the Buyers/Suppliers site to be able to connect to it and post events. It also performs event filtering before notifying the subscribers.

The Event Server on one side performs the task of listening to events from the Event Server of the other sites and forwards these events to its corresponding ETR Server to activate rules.

4.2.2 ETR Server

The ETR Server performs the installation and processing of triggers and rules defined for each site. The ETR Server interacts with the Knowledge Profile Manager and the Event Server. It receives the definitions of rules and triggers input at the Buyer/Supplier site using the graphical user interface provided by the Knowledge Profile Manager. These triggers and rules are translated into internal data structures and program code for run-time processing, respectively. The ETR Server also receives event notifications from the Event Server when events occur. It would look up the internal data structures to identify the proper triggers and schedule the appropriate rules for execution.

4.2.3 Knowledge Profile Manager

The Knowledge Profile Manager (KPM) is responsible for maintaining the knowledge profile of the Buyer/Supplier site that participates in a supply-chain. The knowledge profile of the Buyer/Supplier site contains the events, triggers, and rules that are published by the Buyer/Supplier in the supply-chain.

The Knowledge Profile Manager provides GUI interfaces for defining events, triggers, and rules [PAR99]. Any person on the Buyer/Supplier side with no programming experience can use the GUIs to define events, triggers, and rules by using
forms that contain text boxes and drop-down menus. KPM maintains separate profiles for a user or organization that plays both roles as a Supplier and a Buyer.

4.2.4 Persistent Object Manager

The Persistent Object Manager (POM) consists of two main components: 1) Object-Relational Mapping Engine and 2) XML-Relational mapping engine.

The general-purpose Object-Relational Mapping Engine provides a persistent storage facility for storing information related to the SCM. It also provides a high-level mechanism, in the form of APIs, for programs to store, retrieve, update and delete objects without having to know the internal data structures of the objects. The programs executing on any site use these APIs to perform their database-related operations. The XML-Relational Mapping Engine provides the persistence capability and a filtering mechanism to the Event Server of each business entity’s site. It includes a parser, which maps the filter definitions of the SCM events stored in the form of XML files to their relational representations, and a set of Application Program Interfaces (APIs). POM is implemented on top of an Object-Relational database system called Cloudscape.

To utilize POM for persistent storage at the supply-chain node, a configuration file for that node’s database must be created and stored in the directory of the program that invokes the APIs of POM. This database is then used for storing the supply-chain related information such as Supplier information, Buyer information, Quotes from Suppliers, etc. Figure 14 is a sample of a configuration file.

<table>
<thead>
<tr>
<th>Host</th>
<th>localhost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>9099</td>
</tr>
<tr>
<td>Database</td>
<td>SCMDB</td>
</tr>
</tbody>
</table>

Figure 14. Sample of a Configuration File for the SCM Database
The first parameter specified is the host on which the database is located. In the SCM system, all of the supply-chain related database operations are performed on the SCM server side. Hence, the value of the host parameter is specified as “localhost” since the local applications are the ones that issue the database operations. This parameter can also be a machine name or the IP address of a remote server.

The second parameter is the connection port number. POM uses this port number for connection. No other program can be allowed to use this port number. In the SCM system, 9099 was empty, and was chosen to be the port number.

The final parameter specified is the name of the database. The database is created in the default path specified in the Persistent Object Manager’s settings.

Once a configuration file is specified in the directory of the supply-chain programs utilizing the APIs of POM, no additional measures are needed to create the database. The first time that the SCM database is used, POM checks to see if the specified database in the configuration file exists. If it does not, it creates the database in the specified default path, its log files, and related information. If the database already exists, then POM directly proceeds to perform the requested operation on the SCM database.

4.2.5 Metadata Manager

Buyers or Suppliers decide the appropriate events, triggers, and rules that should be created on their sites to manage their business processes. Once created using the Knowledge Profile Manager, these events, triggers and rules information are stored and maintained by the Buyers/Suppliers sites’ Metadata Managers.
4.2.6 Cost-Benefit Evaluation Server

This Cost-Benefit Evaluation Server (CBES) is based on a Cost-Benefit Decision Model (CBDM) which is explained in a paper by Stanly Su, et al. [SU87]. In this model, the contents of each product specification are divided into two structures: (1) the cost structure, which contains all the attributes to which costs can be assigned (e.g., a specific disk, an additional memory board, etc.) and (2) the preference structure, which contains all the attributes to which preference scores can be assigned subjectively (note: these two sets of attributes may overlap). The structures are separately analyzed to obtain two aggregated values: an aggregated cost value and the global preference score. These two values are then combined to derive a global cost-preference indicator for the specification of a system under evaluation.

CBES and its services are useful for supporting decision-making in Supplier selection, negotiation proposal evaluation, and evaluation of responses for a request for quote or any other type of business specifications that can be defined by a hierarchical structure of attributes and values in which costs and benefits are involved.

4.2.7 Negotiation Server

The core e-services provided by the servers described above have been used to implement the Negotiation Server. The services provided by the ETR and CBES servers were used in an implemented negotiation server. All these servers are replicated and installed at the sites of business organizations that conduct negotiations.

Each negotiation server would carry out a negotiation process on behalf of its client, based on some registered bargaining strategies (managed by an ETR server), product/service constraints (managed by a Constraint Satisfaction Processing (CSP) server) and costs, preference scoring methods, and aggregation functions (managed by
the CBES server). Negotiation proposals and counter-proposals are passed between a pair of Negotiation servers in a bilateral bargaining situation. The Negotiation server uses CSP to match the received proposal against the receiver’s product/service specification. If any violation of constraint is detected, it will post an event that corresponds to the violation to trigger some concession rule to relax the receiver’s constraint and generate a counter-proposal to its counterpart. If no constraint violation is found in the negotiation proposal and the proposal specified several value combinations that are acceptable to the sender of the proposal, the CBES server is used to evaluate these value combinations to determine the best cost-benefit combination to present to the receiver for its acceptance.

4.2.8 Constraint Satisfaction Processing Server

The Constraint Satisfaction Processing Server is responsible for evaluating the constraints given in a negotiation proposal or counterproposal against the pre-registered constraints of a negotiation party. CSP is capable of identifying the constraints that have been violated and posting events to trigger the processing of decision-action rules. A detailed explanation of CSP can be found in [LI01].
CHAPTER 5
SCM IMPLEMENTATION

SCM has been implemented using JDK 1.3, on a Windows NT platform. The Web Server used is Jakarta-Tomcat 3.2.

In SCM implementation, we use the events and the rules to integrate different services such as a negotiation service, a cost-benefit evaluation service and a constraint satisfaction processing service. With the use of the events and the rules any of these services can be called from any of the rules giving a great flexibility to the entities in supply-chain modeling of their business processes. Through the use of rules, an entity in supply-chain can decide when and how it wants to use these services. For example, a Retailer can decide to evaluate all the quotes it receives from the Distributors first, and then later it may want to do a negotiation or it may decide to just accept the quote the Distributor has sent and generate a purchase order. We will describe in later sections how these events and rules are used to integrate the above services.

5.1 Tables in the SCM Database

The SCM system uses the Object-Relational Mapping Engine of the Persistent Object Manager to perform database operations. Therefore, it is necessary to have objects corresponding to relational tables in SCM. To contain the necessary information for different SCM processes, we have developed four objects, each of which corresponds to a relational table in the SCM database. The following table shows the data members in these objects and a brief description of their purposes.
Table 4: Description of Objects in SCM and their Attributes

<table>
<thead>
<tr>
<th>Object/Table Description of Object</th>
<th>Data Members</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quote</strong></td>
<td>prodno</td>
<td>Int</td>
<td>Product Number</td>
</tr>
<tr>
<td></td>
<td>productPrice</td>
<td>Double</td>
<td>Product Price</td>
</tr>
<tr>
<td></td>
<td>qty</td>
<td>Int</td>
<td>Qty required</td>
</tr>
<tr>
<td></td>
<td>QuoteFrom</td>
<td>String</td>
<td>Name of Supplier</td>
</tr>
<tr>
<td></td>
<td>QuoteTo</td>
<td>String</td>
<td>Name of Buyer</td>
</tr>
<tr>
<td></td>
<td>deliveryDate</td>
<td>GregorianCalendar</td>
<td>Delivery Date</td>
</tr>
<tr>
<td></td>
<td>IPAddress</td>
<td>InetAddress</td>
<td>IP address of the sender</td>
</tr>
<tr>
<td><strong>Orders</strong></td>
<td>ProductNo</td>
<td>Int</td>
<td>Product Number</td>
</tr>
<tr>
<td>This object/table contains the information provided by a Buyer while sending order.</td>
<td>Price</td>
<td>Double</td>
<td>Product Price at which the item is ordered</td>
</tr>
<tr>
<td></td>
<td>Qty</td>
<td>Int</td>
<td>Quantity to be purchased</td>
</tr>
<tr>
<td></td>
<td>DeliveryDate</td>
<td>Gregorian Calendar</td>
<td>Delivery date expected</td>
</tr>
<tr>
<td></td>
<td>OrderFrom</td>
<td>String</td>
<td>Buyer’s name</td>
</tr>
<tr>
<td></td>
<td>OrderTo</td>
<td>String</td>
<td>Supplier’s name</td>
</tr>
<tr>
<td><strong>Product</strong></td>
<td>ProductName</td>
<td>String</td>
<td>Name of the product</td>
</tr>
<tr>
<td>This object/table contains the information provided of a product</td>
<td>ProductNo</td>
<td>Int</td>
<td>Product Number</td>
</tr>
<tr>
<td></td>
<td>PurchasingPrice</td>
<td>Double</td>
<td>Purchasing Price for that product</td>
</tr>
<tr>
<td></td>
<td>SellingPrice</td>
<td>Double</td>
<td>Selling Price for that product</td>
</tr>
<tr>
<td><strong>Inventory</strong></td>
<td>ProductNo</td>
<td>Int</td>
<td>Product Number</td>
</tr>
<tr>
<td>This object/table contains the information about inventory of the product.</td>
<td>Qty</td>
<td>Int</td>
<td>Qty available</td>
</tr>
<tr>
<td></td>
<td>ROP</td>
<td>Int</td>
<td>Re-Order-Point for this product</td>
</tr>
<tr>
<td></td>
<td>MaxLevel</td>
<td>Int</td>
<td>Max level for this product</td>
</tr>
</tbody>
</table>
5.2 Classes Used in Implementing SCM

Now we will explain the classes used to implement the SCM system. We will start with the Retailer side classes. The main class on the Retailer side is the RetailerMain.

- **RetailerMain.** This class starts the Retailer system. It listens to the customers for their requests for products. Upon receiving requests for products this class calls the inventory control system algorithm. If the inventory is not enough, then a request for quote is generated for that product. This request for quote is posted to the ETR server using the RFQ (request-for-quote event). Then this class waits for the replies from the Suppliers for a specified time. Replies are posted by Distributors by posting reply-RFQ event. Once the timeout occurs or the Retailer receives the pre-specified number of quotes, it then evaluates these quotes using the Cost-Benefit Evaluation Server. If the received quotes are not up to the expectations of the Retailer, then this class calls the Negotiation Server, which negotiates with the Negotiation Server of a selected Distributor on behalf of the Retailer. If the negotiation succeeds, then the Generate-Purchase-Order method is called which generates a purchase order. Then, this purchase order is sent to the respective Distributor. Then this class waits for the Distributor to ship the order.

- **RetailerInventoryControlSystem.** This class checks current inventory of a product. Upon receiving a product request made to the Retailer, this class’s checkInventory() method is called which takes the product number and quantity as arguments. Then it queries the Persistent Object Manager to get the inventory object for that product and checks whether the quantity requested can be satisfied. This inventory object also stores the Re-Order-Point of this product and the maximum level of this product. Upon receiving this inventory object, this class checks if the inventory goes below the ROP after satisfying the request. If yes, it returns false which starts a new process to post an RFQ event.

- **GeneratePurchaseOrder.** This class generates a purchase order and sends this purchase order to the selected Distributor. This class extracts the IPAddress from the Distributor’s Quote object that was received and sends the Order at that IPAddress.

Now we will discuss about the Distributor side classes

- **DistributorMain.** This class starts the Distributor system. This Distributor Class is activated when an RFQ event is posted by the Retailer System. This class does all the work of calling respective classes when required. After receiving the request for quote, this class calls the Distributor’s inventory control system which checks whether the Distributor has enough inventory to satisfy the received order. If not then it in turn sends a RFQ (Request-for-Quote) to Manufacturers. The
quantity required is calculated in the inventory control system and is used in the Request-For-Quote. Upon receiving the quotes from the Manufacturers, this class evaluates all the quotes in the same way as the Retailer. It then sends the reply to the Request-For-Quote to the Retailer. If the Retailer starts a negotiation with the Distributor, the Distributor in turn starts negotiation with the Manufacturer and, based on this negotiation’s results, the Distributor negotiates with the Retailer. Upon successful negotiations with both the Retailer and the Manufacturer, it waits for the Retailer to send the purchase order. Depending on that purchase order, the Distributor sends a purchase order to the Manufacturer.

- **DistributorInventoryControlSystem.** This class does the same work as RetailerInventoryControlSystem. This class checks the inventory on hand as well as inventory position. Depending on each product’s ROP and max level, this class finds out whether it can satisfy the RFQ sent by the Retailer.

- **EvaluateQuote.** This class does the same work of evaluating the quotes received from Manufacturers as that of the Retailer’s EvaluateQuote. This class is called when either the timeout for receiving quote occurs or the required number of quotes has been received. This class in turn calls the Cost-Benefit Evaluation Server and sends all the quotes to it. The Cost-Benefit Evaluation Server evaluates these quotes and assigns an overall value to each quote. This class then sorts these evaluated quotes in a decreasing order.

- **GeneratePurchaseOrder.** This class is the same as that of the Retailer’s GeneratePurchaseOrder. This class generates a purchase order to be sent to the Manufacturer.

- **SendQuote.** This class sends a reply to Retailer’s Request-For-Quote. This class takes the quote object returned by InventoryControlSystem as input and posts it using a replyRFQ event, which is subscribed by the Retailer.

- **ProcessOrder.** This class does the actual processing of the orders received from the Retailer. Upon receiving an order from the Retailer this class checks whether the inventory is enough to satisfy the order and whether it has already sent an RFQ to Manufacturers. If the inventory is enough, it reduces the inventory by the quantity requested in the Retailer’s purchase order and ships it to the Retailer. If the inventory is not enough, it checks the agreed proposal received from the Negotiation Server during the Distributor’s negotiation with the Manufacturer. It then generates a purchase order and sends it to the Manufacturer.

Manufacturer side classes are the same as that of the Distributor side except that it does not send any RFQ. Here are the Manufacturer side classes

- **ManufacturerMain.** This class starts the Manufacturer’s system. It is activated by the RFQ event posted by the Distributor. Upon receiving the RFQ from the
Distributor, this class calls the InventoryControlSystem on the Manufacturer side. If there is not enough quantity on hand then it calls production department’s job scheduling algorithm. And according to the results returned by the scheduling algorithm this class generates a reply to RFQ and posts it using the replyRFQ event. Then this class waits for the purchase order from the Distributor. After receiving the purchase order, this class calls the ProcessOrder class which processes this order and sends the shipment.

- **InventoryControlSystem.** This class handles the inventory control on the Distributor side. CheckInventory method takes quantity as input and checks this quantity against the product’s inventory. If this inventory is not enough to satisfy the requested quantity, then it calls the scheduling algorithm. According to the results returned by the scheduling algorithm, this class returns the delivery date.

- **SendQuote.** This class is the same as that of the Distributor’s SendQuote object.

- **ProcessOrder.** This class is the same as that of the Distributor’s ProcessOrder object.

The communication between all these classes is through XML documents. While posting an RFQ event, the Quote object is converted into an XML document and sent through the event. Also, while sending a purchase order, it is converted into an XML document and then sent.

### 5.3 Posting of SCM Events

SCM has three events in the present implementation. The method of posting these events is described below:

- **RFQ (Request for Quote).** This event is posted when a Retailer’s or Distributor’s inventory is not enough to satisfy customer’s demands. This event is also posted when the Retailer’s inventory level goes below the Re-Order-Point. This event consists of a Quote Object, which is converted into an XML string. This event helps Retailer to send RFQ to as many Distributors as it wants. This saves Retailer’s time by saving one-on-one contact between Retailer and each Distributor.

- **ReplyRFQ.** This event is posted in reply to the RFQ event. The Distributor or the Manufacturer posts this event after it receives a Request-for-Quote event. This event also consists of a Quote object, which is converted into an XML string.
51

- **TimerEvent.** This event is posted when the timeout to submit a ReplyRFQ occurs. This event is posted by the Timer Event Server. This event is posted on the Buyer side. The Buyer, which sends a Request for quote, decides the actual amount of time that it is willing to wait to receive quotes from Suppliers. The Timer Event Server posts this timer event when the timeout occurs.

5.4 Classes in Retailer’s, Distributor’s and Manufacturer’s Rule Library

As described previously, SCM is designed to allow each business entity to define rules on its own site to implement its business strategies. We will explain some of the classes used in Retailer’s Rule Library.

- **InsertQuote.** This class is executed from the Buyer’s side rule – SaveQuote. Its primary function is to store the quote received from a Supplier in the Persistent Object Database for future reference. It has the following method
  - Public static void insert(String S): This method takes S as a parameter which is an XML document and converts it to a quote object. Then it stores this object into the database using POM.

  This class is used to integrate the ETR server and the POM. With this rule quotes can be stored in persistent database POM and can be used later for negotiation or cost-benefit evaluation.

- **RemoveTimerEntry.** This class is executed when the Buyer receives the required number of quotes before a specified timeout occurs. In that case, the TimerEntry has to be removed from TimerEventServer so that no event is posted when timeout occurs. This class has a method called remove() which does all this work.

- **GetCounter-InsertCounter.** These classes are executed each time a Buyer receives a quote from a Supplier. The counter is stored in the Cloudscape database using POM. Whenever the Buyer receives a quote it calls the GetCounter objects get() method to retrieve the counter, it increases the counter’s value by one and stores the new value to the persistent database by using InsertCounter’s insert() method. This counter object is used to check if the Buyer has received the required number of quotes.

- **EvaluateQuote.** This class is called when either the timeout for receiving quotes occurs or the required quotes have been received. This class is called by the EvaluateQuote rule. This class in turn calls the Cost-Benefit Evaluation server and sends all the received quotes to it. The Cost-Benefit Evaluation server evaluates these quotes and assigns an overall cost-benefit value to each quote. This class then sorts these evaluated quotes in a decreasing order. This class is used to integrate ETR server, Cost-Benefit Evaluation Server and POM. Here POM is used to store the evaluated quotes in sorted order.
• **WakeupDistributor.** This class is called when the Distributor receives a Request-for-Quote from the Retailer. RFQRule calls this class’s wakeup method which activates the Distributor side’s main process.

• **WakeupManufacturer.** This class does the same work as that of the Distributor’s WakeupDistributor class. It is called by RFQRule.
CHAPTER 6
CONCLUSION AND FUTURE WORK

In this thesis, we have presented the concept and the techniques of building an event-trigger-rule based supply-chain management system over the Internet. An Internet-based supply-chain management system, SCM, has been implemented to test and demonstrate the concept and techniques. SCM is constructed by a network of SCM nodes, each of which consists of a Web server, an Event Server, an ETR Server, a Knowledge Profile Manager, a Persistent Object Manager, a Metadata Manager, a Cost-Benefit Evaluation Server, a Timer Server, a Constraint Satisfaction Processing (CSP) Server and a Negotiation Server. By installing the SCM Node at multiple sites, business entities can use various services provided by these servers. SCM offers a number of advantages over the existing supply-chain management systems. First and foremost is the flexibility offered to business entities in defining their own rules according to their own business strategies. Hence there is no need for business entities to hard code their business strategies in software systems. Second, since the rules that control the business activities are installed and processed by the ETR servers which are installed at the business entity’s individual sites, an entity’s privacy and security are safeguarded. Third, SCM’s event, event filtering and event notification mechanisms keep both Buyers and Suppliers better and more timely informed of business events allowing information sharing through the events. Also while sharing information, secrecy of each partner’s confidential data is maintained by allowing it to define its own events. Fourth, our system allows business processes to adapt to the runtime behavior of the system allowing
more flexibility in operation. Fifth, our system can be used in a large supply-chain by replicating SCM nodes for each business entity.

In spite of the above desired features of SCM, some issues for further investigation may be addressed in future work. Currently, business processes are being specified in terms of simple rules and formulas. More complex rules and formulas can be defined and applied to take more factors into consideration. Also, in a supply-chain system environment, a Buyer may negotiate with multiple sellers and a seller may negotiate with multiple Buyers concurrently. The bilateral negotiation discussed in this thesis is just one thread in a multi-lateral negotiation effort. In some negotiation situations, it is possible to carry out a multi-bilateral negotiation by multiple bilateral negotiations. However, due to the added complexity, many additional problems need to be dealt with. The progress and the result of each bilateral negotiation need to be coordinated with those of the other bilateral negotiations. The decision to be made in one step of the negotiation may depend on the progress or result of another. For example, if one Supplier is able to supply a large quantity of parts at a good price, the Buyer may have to terminate some of the on-going negotiations with other Suppliers or to reduce the quantity requested from them. Or, if one or more sellers have decided to make a big concession, the Buyer has the power to concede very slowly in negotiations with other sellers.
REFERENCES


BIOGRAPHICAL SKETCH

Rakesh Lodha was born on April 6, 1976 in Barshi, Maharashtra, India. He received a bachelor’s degree in mechanical engineering, securing a First Class with Honors from the University of Bombay, Bombay, India, in July 1997. From 1997 to 1999, he was a management trainee for Rohit Inc, a supply-chain distributor, India.

He joined the University of Florida in May 2000 to pursue a master’s degree in computer science and engineering. He has been a research assistant and teaching assistant during his studies at the University of Florida.

His research interests include supply-chain management, databases, and e-commerce.