5-1-2006

A supply chain collaboration solution using XQuery

Bharani Govindasamy

A Supply Chain Collaboration Solution Using XQuery

Bharani S. Govindasamy

M.S. (Industrial and Systems Engineering)

Thesis submitted in partial fulfillment of the requirements

for the Master of Science in department of Industrial and Systems Engineering

in the Kate Gleason College of Engineering

of the Rochester Institute of Technology

May 2006

KATE GLEASON COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NY

CERTIFICATE OF APPROVAL

MASTER OF SCIENCE DEGREE THESIS

The M.S. Degree Thesis of Bharani S. Govindasamy has been examined and approved by the thesis committee as satisfactory for the Thesis requirement for the Master of Science Degree in Industrial and Systems Engineering

---

Dr. Sudhakar Paidy

Dept. of Industrial and Systems Engineering

Kate Gleason College of Engineering

Dr. Marcos Esterman

Dept. of Industrial and Systems Engineering

Kate Gleason College of Engineering

Thesis/Dissertation Author Permission Statement

Title of thesis or dissertation: A SUPPLY CHAIN COLLABORATION SOLUTION USING XQUERY

Name of author: _____

Degree: M.S

Program: INDUSTRIAL ENGINEERING

College: KATE GLEASON COLLEGE OF ENGINEERING

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

Print Reproduction Permission Granted:

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

Signature of Author: _____ Date: 05/01/2006

Print Reproduction Permission Denied:

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

Signature of Author: _____ Date: _____

ACKNOWLEDGEMENT

This thesis work was an enjoyable journey made possible with the help of a number of people who provided directions when I found myself lost.

Firstly, I would like to thank Dr. Sudhakar Paidy, for his significant contribution in grooming me as a professional; for his care and support at the times when I needed it most; and for teaching me the importance of "First things First."

I would like to thank Dr. Marcos Esterman, for his valuable comments.

I would like to thank Pradip Chandratre, at the Advanced Systems Integration Lab, for his patience in listening and questioning all my ideas and providing a helping hand in Visual FoxPro programming.

I would like to thank Vilesh Salumkhe, at the Advanced Systems Integration Lab, for helping me with the logistics. Last, but not least, I would like to thank my friends for providing valuable support and comments, from the beginning of my thesis.

DEDICATION

To my Mom,

For her unconditional love

TABLE OF CONTENTS

Abstract

Enterprises in a collaborative supply chain share information for quick and intelligent decision making. Collaborative Planning, Forecasting and Replenishment (CPFR), a business model developed by an organization called Voluntary Inter-industry Commerce Standards, defines a set of business processes to collaborate on planning, forecasting and replenishment activities to arrive at a common order. This involves sharing sales forecast, order forecast and promotion information among business partners. The Internet has paved the way for using Web Services to share this information as XML data in real time and to make it available across enterprises. Enterprises need to access this information and integrate it with the information available in their databases, Excel sheets and XML applications.

This thesis discusses the development of an information architecture using XQuery (a declarative query language by W3C) for aggregating data from disparate data, and evaluates XQuery as a tool to enable information integration for supply chain collaboration. For this purpose, several CPFR scenarios are identified where two retailers share their sales forecast, order forecast and promotion forecast information with the manufacturer. A prototype application is developed which can fetch data from disparate data sources such as Web Service, XML databases and CSV files, aggregate it, and use it to identify exceptions in forecasts and calculate the key performance indicators (KPI). These KPIs are then presented as an executive dashboard using Visual FoxPro. This Dashboard presents information graphically and enables executives to track the performance of the supply chain, and hence enables them to make quick and intelligent decisions.

Introduction

The supply chain is the series of links and shared processes that exist between suppliers and customers. These links and processes involve all activities from the acquisition of raw materials to the delivery of finished goods to the end consumer. Raw materials enter into a manufacturing organization via a supply system and are transformed into finished goods. The finished goods are then supplied to consumers through a distribution system. Generally, several companies are linked together in this process, each adding value to the product as it moves through the supply chain. Since the value addition for a product does not rely on just the manufacturer, the concept of supply chain management assumes significance [12].

We can define supply chain management as the things we do to influence the behavior of the supply chain and get the results we want. Thus we can define supply chain management as "The systemic, strategic coordination of the traditional business functions and the tactics across these business functions within a particular company and across businesses within the supply chain, for the purposes of improving the long-term performance of the individual companies and the supply chain as a whole" [1].

Traditional supply chain management techniques assumed that the information would be available for an immediate supplier and customer in a business. This gave way to a linear supply chain where information was shared only between a stakeholder's immediate participants in the supply chain. This resulted in the bullwhip effect, which refers to the demand variability increasing as one moves up the supply chain away from the retail customer. This means that small changes in consumer demand can result in large variations in orders placed upstream. Studies on managing the bullwhip effect pointed to the need for a collaborative supply chain where all the participating business partners share information and make it available to all the other business partners [2]. This would mean that information visibility across the supply chain enables companies to notice the variations in demand as a result of logistics issues, plant shutdowns, promotions, etc., in real time, and hence to plan for managing the variations in advance [9].

Several open standards organizations have defined business processes for supply chain collaboration. The Voluntary Interindustry Commerce Standards (VICS) organization defines a standard for combining the task of planning, forecasting and replenishment for increased efficiency in a supply chain. Their Collaborative Planning, Forecasting, and Replenishment (CPFR®) model formalizes the processes between two trading partners used to agree upon a joint plan and forecast, monitor success through replenishment, and recognize and respond to any exceptions. CPFR builds upon Efficient Consumer Response (ECR) principles including Vendor-Managed Inventory (VMI), Jointly-Managed Inventory (JMI) and Continuous Replenishment (CRP) [7]. These process standards involve exchange of information between enterprises.

Information exchanges assist in transporting data across the supply chain to be used by all business partners. Traditional information exchanges used electronic data interchange (EDI) technology that enabled businesses to build systems and share data in a specified format agreed upon by all the business partners. The information exchange was carried out through networks such as WAN. The disadvantages of using the EDI were that it was too expensive to implement and the business partners were forced to build a separate system for each business partner they interacted with. Also, the business software applications across the enterprises used disparate software components that made the task of integrating the information systems across the enterprises difficult [3].

As the supply chain grew larger, enterprises found it difficult and costly to implement EDI, and there was a need for data and communication standards for information exchange [3]. This need for a standard message format and communication medium paved the way for the emergence of Web Services that advocated interoperability between software systems [4].

Web Services are software programs that can interact with the other software programs via the Internet using the open standards for data and communication. Web Services are loosely coupled, which means integrating applications that are built using disparate components is easy, and hence data and application integration can be effectively managed [4]. Web Services use XML as the standard for sharing data. XML stands for Extensible Mark-up Language (XML),

which was developed as a means to describe data and has now developed into a data standard as a result of its excellent support in transporting data through the Internet [13].

As companies choose to implement Web Services to enable supply chain visibility, the need has arisen to integrate this information (as XML) that is available from the Web Services, at the business partner end, and the enterprise information that is available in the databases, at the stake holder's end, in real time or near real time for the businesses to take intelligent and quick actions on purchasing and delivery issues. Enterprise information integration (EII) has grown into a business strategy and deals with this task of integrating disparate data sources such as XML from Web Services and relational data from databases. Enterprise information integration is focused on the integration of data from multiple systems into a unified, consistent and accurate representation geared toward the viewing and manipulation of that data. The type of data has to be real-time, which means that the information from the different data sources has to be processed in real time and made available as and when it is needed [5].

Enterprises today rely on information from disparate data sources. Collaborative supply chains today are growing in complexity as more and more business partners become part of the product supply chain and the information needs of the different partners are ever-increasing [9]. The widespread adoption of XML has profoundly altered the way the information is exchanged within and between enterprises. With a vast amount of data being published in XML format by multiple sources, the need has arisen for an easy and efficient means of extracting and manipulating this information for managing the supply chain. Enterprise information integration (EII) has become more than just a buzzword; it has grown into an enterprise strategy to integrate information and make it available in real time or near real time [11].

Internet-enabled enterprises are starting to use XML for information-sharing as a result of XML's excellent support for transporting data in a structured format over the Internet [10]. An enterprise that collaborates with business partners requires that the information available from disparate data sources such as XML and relational databases be integrated and presented [11].

Technologies such as JDOM are built on existing programming languages such as Java and are capable of manipulating data from different data sources [6]. Software programs can be written to access the data from different data sources using application programming interfaces (API), each written for the purpose of accessing and processing data from a specific data source. The other type of languages that perform the task of accessing and processing XML are called query languages. Unlike programming languages that are descriptive (where we specify how we want something done), these are declarative languages, which means we just have to specify what we want [6]. World Wide Web Consortium (W3C), a non-profit organization for developing open standards for communication over the Web, has come up with a declarative query language called XQuery. XQuery holds the promise of efficiently processing (i.e., joining and sorting) information from disparate sources [8]. Although XQuery is not yet a specification, its working draft version enables us to explore the possibility of enterprise information integration in detail. Since information integration is a paramount task in supply chain collaboration activities, we will focus our study on evaluating XQuery for supply chain collaboration.

This thesis intends to study XQuery and evaluate it as a tool for supply chain collaboration. For the purpose of this study, we will consider the supply chain collaboration scenarios defined by Voluntary Interindustry Commerce Standards Association (VICS) [7]. VICS is an open industry organization chartered to promote cross-industry commerce standards for supply chain collaboration. Their supply chain collaboration methodology is called Collaborative Planning Forecasting and Replenishment (CPFR), and VICS describes the business processes that are involved in CPFR. The cases used for this study are derived from the CPFR Process Model defined by VICS.

Brief overviews of the chapters to follow are mentioned below.

Chapter 3 defines a supply chain and gives an overview of the basic features of a supply chain. The various types of supply chain are explained. The need for managing a supply chain is explained and definitions for supply chain management are discussed. A brief explanation of collaborative SCM is also discussed.

Chapter 4 discusses the concepts in CPFR. The 9-Step process model is explained, which covers all parts of a CPFR model, including planning, forecasting and replenishment. The key performance indicators (KPI), which are metrics to measure the performance of CPFR, are discussed. The KPIs used in this thesis, such as order forecast accuracy, sales forecast accuracy, promotion forecast accuracy and out of stock frequency, are explained. Executive dashboards are graphical user interfaces which are used to graphically represent the key performance indicators in CPFR. An executive dashboard assists the business partners in a supply chain to monitor the performance of the supply chain in real time and helps in decision making.

Chapter 5 discusses the core XML technologies that are used in this thesis. The emergence of XML opened up a myriad of technologies which can be used for supply chain collaboration. XML Schema, a W3c specification, is a standard to define the structure of an XML document. This chapter discusses XML Schema and compares it with DTD, another W3C specification. Web Services are built upon open standards such as XML, UDDI, WSDL and SOAP. A brief introduction to Web Services and its components are discussed. XQuery, the programming language used for XML aggregation, is discussed. A comparison is made between XQuery and XPATH, a predecessor to XQuery, and the advantages of using XQuery with respect to this thesis are explained.

Chapter 6 explains the various scenarios in CPFR that use the new architecture. Four scenarios are identified, each varying according to the data sources and KPIs. Each scenario is explained along with the data sources used, the processing that is done and the resulting data that is obtained. The resulting data contains the key performance indicators that are used in executive dashboards.

Chapter 7 explains the procedure to develop a supply chain dashboard using XQuery and Visual FoxPro. A sample scenario is used to explain the procedure to identify the data sources, configure the data sources, develop a target XML Schema, develop an XQuery code for data aggregation and develop a dashboard using Visual FoxPro.

1. Supply Chain Management—An Overview

As the business world changes, companies are faced with a wealth of evolving challenges. First they have to cope with constant innovation and shorter product life cycles. Plus, ever-increasing competition forces them to move with greater speed while also keeping costs down. And rising customer expectations mean they have to deliver higher levels of service and individualized products and services. This growing list of challenges is often more than a single company can meet on its own. As a result, more and more companies are teaming up with alliance partners so they can focus on their own core competencies and have access to the full range of resources they need to meet their customers' demands. This means that the companies need to be able to share information quickly within their own enterprise and partners. The field of supply chain management is evolving to address the issues across the supply chain in an organization [1].

As supply chains encompass the companies and the business activities needed to design, make, deliver, and use a product or service, businesses depend on their supply chains to provide them with what they need to survive and thrive. Every business fits into one or more supply chains and has a role to play in each of them. The term "supply chain management" arose in the late 1980s and came into widespread use in the 1990s. Prior to that time, businesses used terms such as "logistics" and "operations management" instead [9].

The supply chain is the series of links and shared processes that exist between suppliers and customers. These links and processes involve all activities from the acquisition of raw materials to the delivery of finished goods to the end consumer. Raw materials enter into a manufacturing organization via a supply system and are transformed into finished goods. The finished goods are then supplied to consumers through a distribution system. Generally, several companies are linked together in this process, each adding value to the product as it moves through the supply chain [9].

1.1 Basic Features of a Supply Chain [1]

Most supply chains exhibit these basic characteristics:

• The supply chain includes all activities and processes used to supply a product or service to a final customer.
• Any number of companies can be linked in the supply chain.
• A customer can be a supplier to another customer so the total chain can have a number of supplier-customer relationships.
• While the distribution system can be direct from supplier to customer, depending on the products and markets, it can contain a number of distributors such as wholesalers, warehouses, and retailers.
• Products or services usually flow from supplier to customer. Likewise, design and demand information usually flows from customer to supplier. (Physical products move "downstream," while demand information flows "upstream.")

Figure 1.1 describes the elements of a supply chain. A simple supply chain encompasses the basic trading partners of a supply chain, whereas the extended supply chain includes the service providers, such as third party logistics providers, who influence a supply chain in one way or another.

Some other definitions of a supply chain are offered below:

• "A supply chain is the alignment of firms that bring products or services to market." [6]

• "A supply chain consists of all stages involved, directly or indirectly, in fulfilling a customer request. The supply chain not only includes the manufacturer and suppliers, but also transporters, warehouses, retailers, and customers themselves." [7]

• "A supply chain is a network of facilities and distribution options that performs the functions of procurement of materials, transformation of these materials into intermediate and finished products, and the distribution of these finished products to customers." [8]

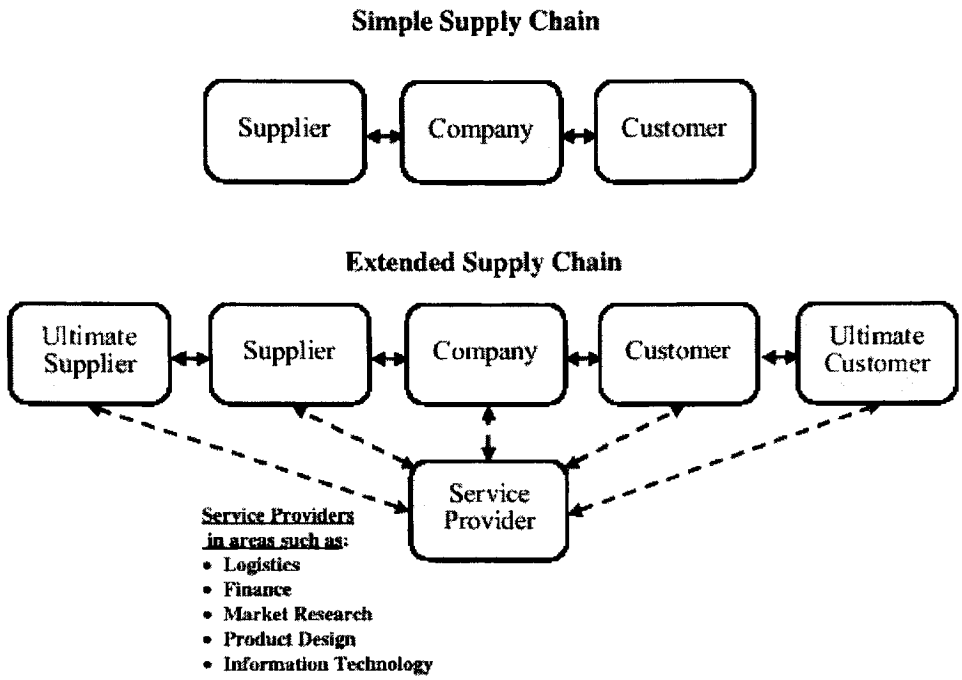

Figure 1.1. Elements of a supply chain [1].

1.2 Types of Supply Chains [1]

A supply chain can be classified by the demand pattern of the products that come out of a supply chain into a (Physically) efficient (Figure 1.2) or a (Market) responsive (Figure 1.3) supply chain. This brings us into classifying products into Functional and Innovative products, both having varying demand patterns. This categorization is very useful as it helps determine the appropriate supply-chain design to support differing demand patterns. Even though this classification is simple, it is very powerful in explaining the mismatches between product attributes and supply-chain design [1].

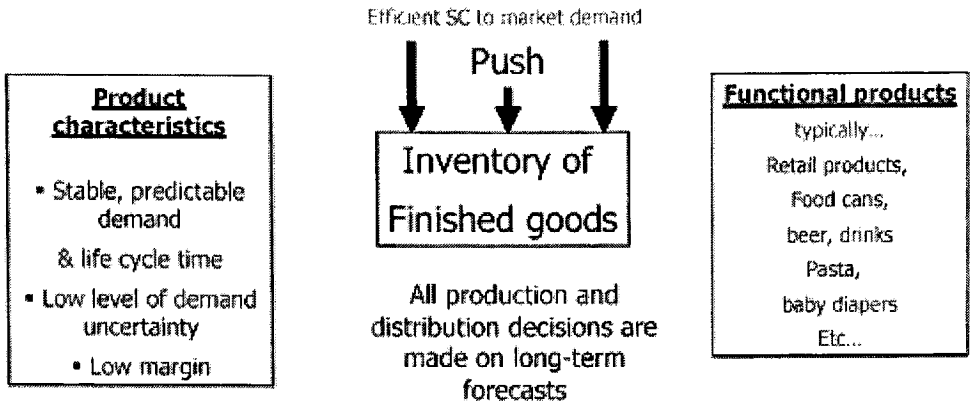

Figure 1.2. An efficient supply chain example: made-to-stock environment [1].

Functional products are commodities that have predictable demand patterns. Some good examples of functional products are light bulbs and toothpaste; these commodities have many substitutes and typically offer low margins for the retailer. They have long lead times to order from suppliers, rare stock-out situations, and usually insignificant price markdowns (unless it is part of a pricing promotion). Innovative products, however, have relatively high margins, very few substitutes (if any), lead times in days from suppliers, frequent stockout situations, drastic markdowns to clear excess inventory, and unpredictable demand. Some good examples of innovative products include Beanie Babies, Furbies, and most recently, Pokemon toys. These products are innovative because no historical scanner data exists to project future sales, as is the case for functional products. Thus, innovative and functional products' characteristics lay the foundation for supply-chain system design [1].

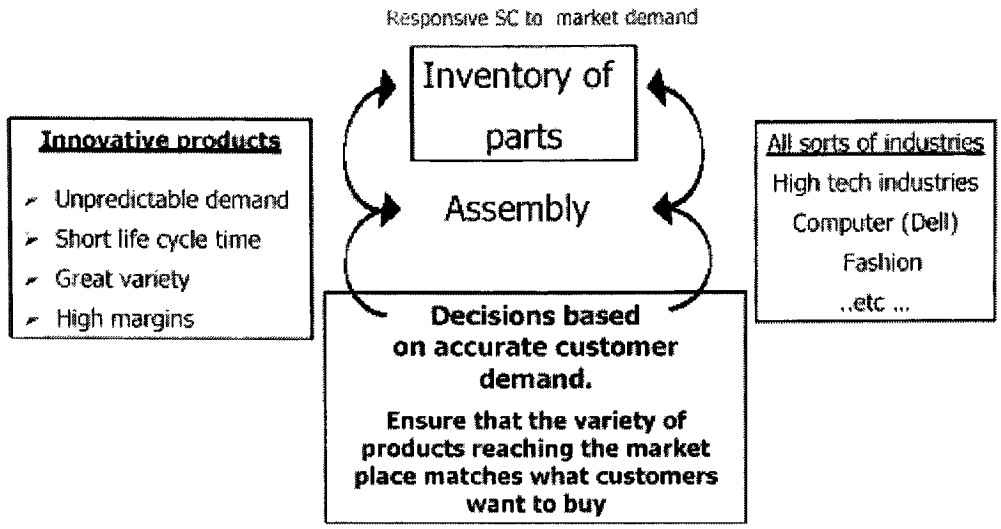

Figure 1.3. A responsive supply chain example: made-to-order environment [1].

Primarily, cost and quality drive supply-chain design for functional products. The entire supply chain's focus is toward reducing costs and getting the right product to the right place, at the right time. Inventory buildups at any point in the supply are not supposed to occur.

Innovative-product supply chains are geared toward supplying products with minimal lead times. This focus on lead times lets retailers respond better to spikes in demand, minimize forced markdowns, and avoid obsolete inventory costs.

The characteristics of the two types of supply chains are important, as they lead to different types of relationships with supply-chain partners. (Table 1.1 and Table 1.2)

Table 1.1 *Selecting a Supply Chain Based on Product Demand Pattern* [1]

	Functional Products	Innovative Products
Efficient Supply Chain	MATCH	MISMATCH
Responsive Supply Chain	MISMATCH	MATCH

Table 1.2 *Characteristics of Efficient and Responsive Supply Chains* [1]

	Physically Efficient Process	Market Responsive Process
Primary purpose	Supply predictable demand efficiently at lowest possible cost	Respond quickly to unpredictable demand to minimize forced markdowns and obsolete inventories
Manufacturing Focus	Maintain high average utilization rate	Deploy excess buffer capacity
Inventory strategy	Generate high returns and minimize inventory throughout the supply chain	Deploy significant buffer stocks of parts or finished goods
Lead-time focus	Shorten lead times as long as it does not increase cost	Invest aggressively in ways to reduce lead times
Approach to choosing suppliers	Select primarily for cost & quality	Select primarily for speed, flexibility and quality

If this is what a supply chain is, then we can define supply chain management as the things we do to influence the behavior of the supply chain and get the results we want.

1.3 Supply Chain Management [1]

Supply chain management is the active management of supply chain activities to maximize customer value and achieve a sustainable competitive advantage. It represents a conscious effort by the supply chain firms to develop and run supply chains in the most effective and efficient ways possible. Supply chain activities cover everything from product development, sourcing, production, and logistics, to the information systems needed to coordinate these activities.

Supply chain management can be further defined as:

"The systemic, strategic coordination of the traditional business functions and the tactics across these business functions within a particular company and across businesses within the supply chain, for the purposes of improving the long-term performance of the individual companies and the supply chain as a whole." [9]

There is a difference between the concept of supply chain management and the traditional concept of logistics. Logistics typically refers to activities that occur within the boundaries of a single organization, while supply chains refer to networks of companies that work together and coordinate their actions to deliver a product to market. Also, traditional logistics focuses its attention on activities such as procurement, distribution, maintenance, and inventory management. Supply chain management acknowledges all of traditional logistics and also includes activities such as marketing, new product development, finance, and customer service [1].

Taken individually, different supply chain activities often have conflicting needs. For instance, maintaining high levels of customer service requires high levels of inventory, but then the requirement to operate efficiently calls for reducing inventory levels. It is only when these requirements are seen together as parts of a larger picture that ways can be found to effectively balance their different demands [1].

Effective supply chain management requires simultaneous improvements in both customer service levels and the internal operating efficiencies of the companies in the supply chain. Customer service at its most basic level means consistently high order fill rates, high on-time delivery rates, and a very low rate of products returned by customers for whatever reason. Internal efficiency for organizations in a supply chain means that these organizations get an attractive rate of return on their investments in inventory and other assets, and that they find ways to lower their operating and sales expenses [2].

As shown in Figure 1.4, the following are five basic components for supply chain management [1].

1. Plan—This is the strategic portion of supply chain management. You need a strategy for managing all the resources that go toward meeting customer demand for your product or service. A big piece of planning is developing a set of metrics to monitor the supply chain so that it is efficient, costs less and delivers high quality and value to customers.

2. Source—Choose the suppliers that will deliver the goods and services you need to create your product or service. Develop a set of pricing, delivery and payment processes with suppliers and create metrics for monitoring and improving the relationships. And put together processes for managing the inventory of goods and services you receive from suppliers, including receiving shipments, verifying them, transferring them to your manufacturing facilities and authorizing supplier payments.

3. Make—This is the manufacturing step. Schedule the activities necessary for production, testing, packaging and preparation for delivery. As the most metric-intensive portion of the supply chain, measure quality levels, production output and worker productivity.

4. Deliver—This is the part that many insiders refer to as "logistics." Coordinate the receipt of orders from customers, develop a network of warehouses, pick carriers to get products to customers and set up an invoicing system to receive payments.

5. Return—The problem part of the supply chain. Create a network for receiving defective and excess products back from customers and supporting customers who have problems with delivered products.

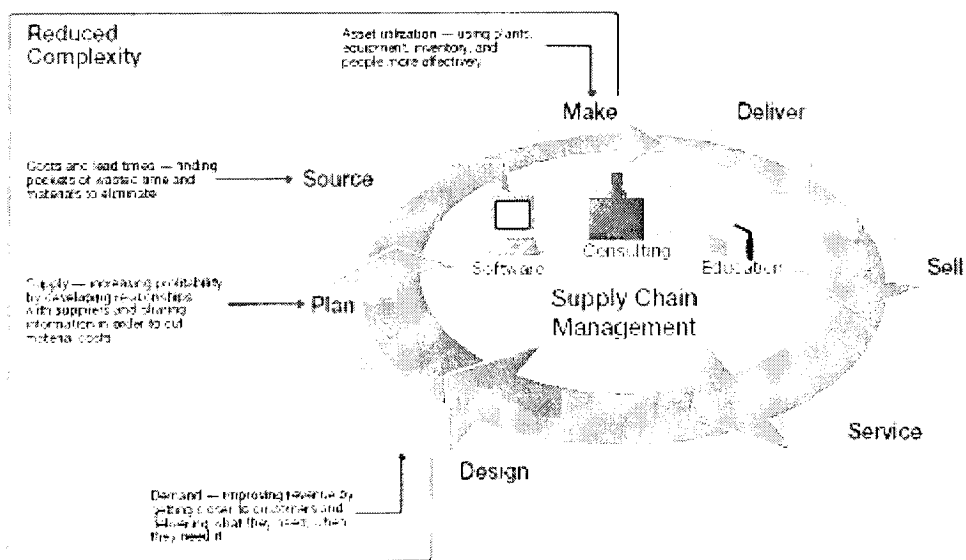

Figure 1.4. Supply chain management life cycle [1].

There is a basic pattern to the practice of supply chain management. Each supply chain has its own unique set of market demands and operating challenges, and yet the issues remain essentially the same in every case. Companies in any supply chain must make decisions individually and collectively regarding their actions in five areas (see Figure 1.5) [1].

1.Production—What products does the market want? How much of which products should be produced, and by when? This activity includes the creation of master production schedules that take into account plant capacities, workload balancing, quality control, and equipment maintenance.

2. *Inventory*—What inventory should be stocked at each stage in a supply chain? How much inventory should be held as raw materials, semifinished, or finished goods? The primary purpose of inventory is to act as a buffer against uncertainty in the supply chain. However, holding inventory can be expensive, so what are the optimal inventory levels and reorder points?

3. *Location*—Where should facilities for production and inventory storage be located? Where are the most cost efficient locations for production and for storage of inventory? Should existing facilities be used or new ones built? Once these decisions are made, they determine the possible paths available for product to flow through for delivery to the final consumer.

4. *Transportation*—How should inventory be moved from one supply chain location to another? Air freight and truck delivery are generally fast and reliable but they are expensive. Shipping by sea or rail is much less expensive but usually involves longer transit times and more uncertainty. Stocking higher levels of inventory must compensate for this uncertainty. When is it better to use which mode of transportation?

5. *Information*—How much data should be collected and how much information should be shared? Timely and accurate information holds the promise of better coordination and better decision making. With good information, people can make effective decisions about what to produce and how much, about where to locate inventory and how best to transport it.

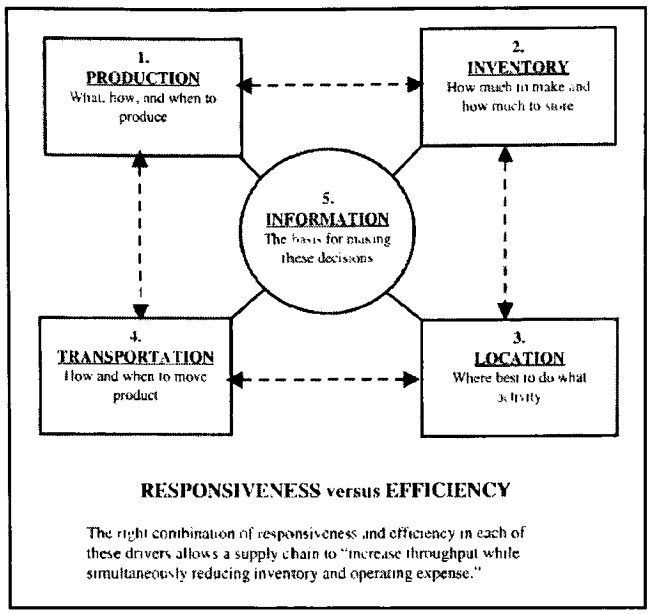

Figure 1.5. Supply chain drivers [1].

1.4 Why Do We Need to Manage a Supply Chain?

An unmanaged supply chain is not inherently stable. Demand variability increases as one moves up the supply chain away from the retail customer, and small changes in consumer demand can result in large variations in orders placed upstream. Eventually, the network can oscillate in very large swings as each organization in the supply chain seeks to solve the problem from its own perspective. This phenomenom is known as the bullwhip effect and has been observed across most industries, resulting in increased cost and poorer service (see Figure 1.6).

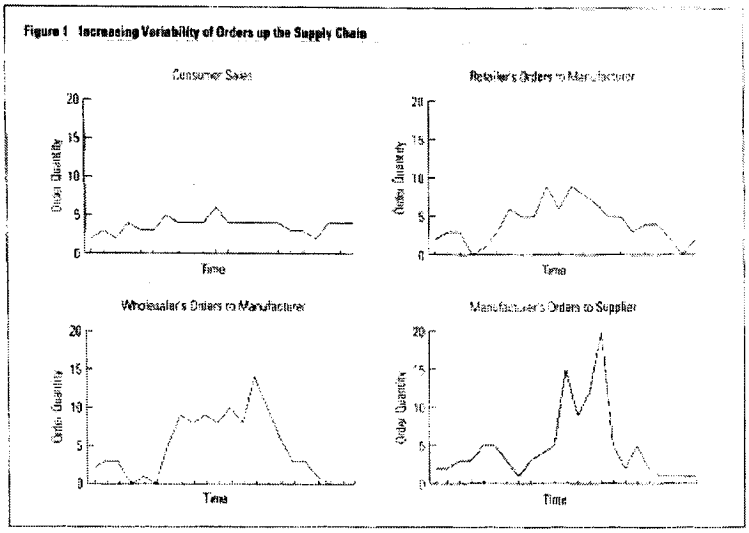

Figure 1.6. Bullwhip effect—variable demand across a supply chain [2].

Bullwhip Effect [2]

Sources of variability include such factors as demand variability, quality problems, strikes or plant fires. Variability coupled with time delays in the transmission of information up the supply chain, and time delays in manufacturing and shipping goods down the supply chain, create the bullwhip effect. All of the following can contribute to the bullwhip effect:

1. Overreaction to backlogs
2. Neglecting to order in an attempt to reduce inventory
3. No communication and coordination up and down the supply chain
4. Delay times for information and material flow
5. Order batching—larger orders result in more variance. Order batching occurs in an effort to reduce ordering costs, to take advantage of transportation economics such as full truck load economies, and to benefit from sales incentives. Promotions often result in forward buying to benefit more from the lower prices.
6. Shortage gaming: customers order more than they need during a period of short supply, hoping that the partial shipments they receive will be sufficient.

7. Demand forecast inaccuracies: everybody in the chain adds a certain percentage to the demand estimates. The result is no visibility of true customer demand.
8. Free return policies

The bullwhip effect can be effectively mitigated only through measures that combine Information Sharing, Channel Alignment and Operational Efficiency. Table 1.3 discusses ways to overcome the causes of the bullwhip effect.

Table 1.3 *Managing the Bullwhip Effect* [2]

Causes	**Information Sharing**	**Channel Alignment**	**Operational Efficiency**
Demand Forecast Updating	* Visibility of End User demand data across the supply chain	* Vendor Managed Inventory (VMI) * Consumer Direct	* Lead-time reduction * Echelon-based inventory control
Order Batching	* Visibility of End User demand data across the supply chain * Order processing costs reduction (e.g., through Computer/ Internet-based Order Management Systems)	* Logistics outsourcing (e.g. 3PLs) * Discounts for truck-load assortments (of different products) * Delivery Consolidation	* Order processing cost reduction (e.g., through Computer/ Internet-based Order Management Systems)
Forward Buying	* Visibility of End User demand data across the supply chain	* Continuous Replenishment Program (CRP) * Stabilize prices through Every Day Low Price (EDLP: selling price) and Every Day Low Cost (EDLC: buying price)	* Activity Based Costing (ABC) to clearly analyze inventory costs, handling costs, transportation costs, etc. and compare them with product price promotion benefit
Shortage Gaming	* Share sales, capacity and inventory data	* Define cancellation policies	

1.5 ERP vs SCM [13]

The differences between ERP systems (e.g. SAP, Baan, People soft) and SCM systems (e.g., I2, Manugistics) have been subject to intense debate. One reason for this is that the ERP vendors are adding more SCM functionality to their products while SCM vendors are also expanding their functionality, encroaching on the area handled by the ERP vendors. With the vendors of ERP systems and SCM systems adding more and more functionality, the differences between them have been blurring. For example, major ERP vendors are introducing advanced planning and optimization as an integrated component (also a component in SCM) of their system.

A few significant differences between a ERP and a SCM system are shown in Table 1.4.

Table 1.4 *Comparison of ERP and SCM Systems* [13]

Point of comparison	**ERP**	**SCM**
Comprehensive	Yes, covers many more areas than SCM	Relatively less
Complexity	Highly complex	Relatively less complex
Sourcing tables	Relatively static	Dynamic
Handling of constraints	In an ERP system, all the demand, capacity and material constraints are considered in isolation of each other	Simultaneous handling of the constraints
Functionality	Relatively less dynamic as they are mainly concerned with transaction processing and have more jobs to do	Can perform simulations of adjustments with regard to the constraints dynamically in real time
Speed of processing requests	Relatively slower	Faster

ERP's Importance in SCM

Traditionally ERP tools were not considered for SCM and as a result, the information flow between various members of the supply chain was slow. This was because until the late 1990's the concentration of organizations was on improving the internal efficiency alone. Therefore, ERP systems also supported only such functionalities, and the systems across the supply chain were disparate [14].

The organizations however, soon realized that although internal efficiency is important, its benefit would be limited unless complemented by increased efficiency across the supply chain. They also realized that seamless flow of real-time information across the supply chain was a key to success in the emerging market scenario characterized by galloping advancements of technology, shorter product life cycles, etc. Therefore, organizations started integrating ERP applications with SCM software. This ensures that the efficiency is achieved across the supply chain and there is a seamless flow of information. In this scenario ERP becomes a vital link in the integrated supply chain as it serves as the integrated planning and control system [14].

In summary, ERP applications help in effective SCM in the following ways [5]:

1. Share data: They can create opportunities to share data across supply chain members, which can help managers in making better decisions. They also provide wider scope to mangers of supply chains by making available much broader information.

2. Real-time information: ERP systems can provide real-time information, which can be of great help in making supply chain decisions. For example, ordering raw materials can be based on the inventory details provided by the ERP systems.

1.6 Expectations from Implementing SCM in an Enterprise [9]

- Minimize the cycle time in receiving projected and actual demand information.
- Establish the monitoring of actual demand for product to as near a real-time basis as possible.
- Understand product demand patterns at each stage of the supply chain.
- Minimize or eliminate information queues that create information flow delays.

- Eliminate inventory replenishment methods that launch demand lumps into the supply chain.
- Eliminate incentives for customers that directly cause demand accumulation and order staging prior to a replenishment request, such as volume transportation discounts.
- Provide vendor-managed inventory (VMI) services by collaboratively planning inventory needs with the customer to projected end-user demand; then, monitor actual demand to fine-tune the actual VMI levels. (Note: VMI can increase sales and profits especially in industries where buyers can go to alternative sources if you or your distributor stock-out.)

Some companies have reaped great benefits from implementing SCM across their enterprises. Table 1.5 lists the percentage improvement for key indicators that measure the performance of a supply chain.

Table 1.5 *Benefits from an Integrated SCM*

Delivery Performance	16%-28% Improvement
Inventory Reduction	25%-60% Improvement
Fulfillment Cycle Time	30%-50% Improvement
Forecast Accuracy	25%-80% Improvement
Overall Productivity	10%-16% Improvement
Lower Supply Chain Costs	25%-50% Improvement
Fill Rates	20%-30% Improvement
Improved Capacity Realization	10%-20% Improvement

1.7 Supply Chain Operations Reference [SCOR] Model

In collaboration with 69 members consisting of manufacturers, logistics / distribution service providers and software solutions suppliers, the Supply Chain Council introduced Supply Chain Operations Reference-model (SCOR) (see Figure 1.7). A number of companies have pooled their real-world supply chain experiences to build a flexible framework and a common language that can help them improve their supply chain internally and externally [4].

The model defines common supply chain management processes and matches them against "best practices." It provides companies with a powerful tool in improving supply chain

operations. It allows manufacturers, suppliers, distributors and retailers with a framework to evaluate the effectiveness of their supply chain operations and to target and measure specific process operations.

The SCOR model was designed to enable companies to communicate, compare and learn from competitors and companies both within and outside of their industry. It not only measures supply chain performance but also effectiveness of supply chain reengineering [4].

The Supply Chain Operations Reference-model (SCOR) is a process reference model. A Process Reference Model helps organizations capture the "as-is" state of a process with the objective to achieve the desired "to-be" future state. Further, it allows an organization to quantify its operational performance, and to establish internal targets based on "best-in-class" results in similar companies. It describes standard management processes, exploring the relationship among different processes. Finally, it characterizes the management practices and software solutions that result in "best-in-class" performance. Thus a process reference model integrates the concepts of business process reengineering, benchmarking, and process measurement into a cross-functional framework.

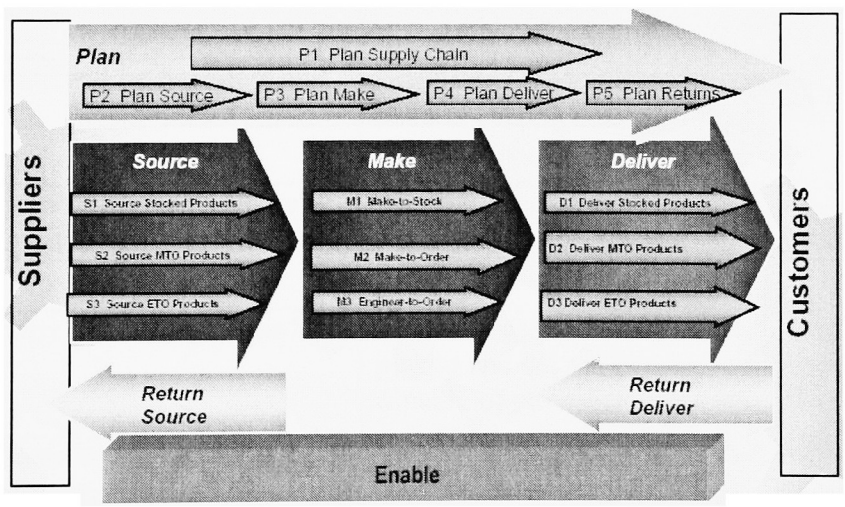

Figure 1.7. Supply Chain Operations Reference [SCOR] model.

1.8 Future of SCM [17]

A lot of activity is going on in the supply chain field that involves implementing supply chain management for planning, execution and collaboration across the enterprise and between enterprises. A few predictions regarding 21st-century supply chain management are tabulated in Table 1.6 [5].

Predictions	Details
Performance will be judged by systems and flows.	Need to improve ways to accelerate the flow.
Artificial intelligence will play a central role in all supply chain activities.	Computer-aided interaction, voice recognition, robotics and expert systems will find more users
Supply chains will be branded and marketed.	Supply chain expertise will be marketed as the products marketed today.
Liability hassles will be alleviated by simplified release-rate pricing.	Legal liability differences between transportation and warehouse service providers will be resolved, as they would move into risk-sharing mode by getting into released-rate pricing structure
Logistics Operations will be paperless and "near labor-less."	Computer-directed machines will handle manual jobs, leaving analytical jobs to human workers
Worker education and training will largely be a corporate responsibility.	To assure steady flow of capable workers, corporations will assume responsibility to train their employees.
Two types of carriers will emerge: line haul and "last mile."	Line haul carriers will contract out all pickup and delivery to last mile carriers capable of providing 24-hr service.
The federal government will be forced to rebuild the transportation infrastructure.	The rail and road infrastructure will deteriorate, forcing the federal government to intervene.

Table 1.6 *Predictions for the 21st Century in Supply Chain Management Field* [17]

1.9 Collaborative SCM

Opportunities for collaboration among business partners will vary depending upon an organisation's prospective role in the supply chain. Collaboration enables partners to jointly gain a better understanding of future product demand and to implement more realistic programmes to satisfy that demand. The three major types of collaborative relationships are described below.

1) Manufacturing/supplier collaboration

Close collaboration among supply chain partners can help to align the parties and then enhance the value of the network's combined activities. Collaborating with suppliers, manufacturers will derive benefits in such key activities as new product development, order fulfilment, and capacity planning. Collaborative product development enabled by sharing and modifying design documents will help manufacturers develop products better and faster. Similarly, co-ordinating all tier-supplier production schedules will help ensure that future material needs are satisfied. This, in turn, results in improved order fulfilment and increased capacity utilisation.

2) Manufacturer/customer collaboration

The collaborative opportunities between manufacturers and customers (such as wholesale-distributors and retailers) center on demand planning and inventory replenishment. The focus is on jointly developing an understanding of demand at the point of consumption, followed by creation of a mutually agreed replenishment plan. This approach helps to ensure that consumer requirements are met efficiently. To collaborate on demand planning successfully, business partners need to share and modify each other's demand plans and forecasts electronically. Importantly, each partner needs to understand and electronically share its promotional plans. Once demand plans and forecasts are in place, replenishment plans designed to assume adequate product availability would be jointly developed.

3) Collaboration with third party and fourth party logistics providers

Collaboration between companies and third party logistics (3PL) providers will focus on joint planning of logistics activities. With regard to transportation services, collaboration will

improve equipment utilisation by enabling the consolidation of inbound, interfacility, and outbound shipments among business partners. This can be accomplished through electronic sharing of information on shipment plans and availability of transportation resources. Packaging is another potential area for logistics collaboration. Collaboration with 3PLs providing distribution centre (DC) services would focus on the productive use of facilities, labor and equipment. This involves electronic sharing of inventory replenishment plans so that receipts do not overload a DC's receiving function or storage capacity. Electronic visibility into the availability of distribution centre resources would support this type of collaboration.

As companies started to implement supply chain collaboration, there was a need to have a standard business process where data and communication for collaboration are well defined and accepted by business partners. Open standards for supply chain collaboration have evolved, targeting collaboration in the supply side or demand side of the supply chain. One such standard, Collaborative Planning, Forecasting and Replenishment (CPFR) is discussed in the next chapter.

2. Collaborative Planning, Forecasting and Replenishment (CPFR)

Several open standards organizations have defined business processes for supply chain collaboration. The Voluntary Interindustry Commerce Standards (VICS) organization defines a standard (see Figure 1) for combining the task of planning, forecasting and replenishment for increased efficiency in a supply chain. Their Collaborative Planning, Forecasting, and Replenishment (CPFR) model formalizes the processes between two trading partners used to agree upon a joint plan and forecast, monitor success through replenishment, and recognize and respond to any exceptions. CPFR builds upon Efficient Consumer Response (ECR) principles including Vendor-Managed Inventory (VMI), Jointly-Managed Inventory (JMI) and Continuous Replenishment (CRP). These process standards involve exchange of information between enterprises [7].

CPFR can be defined as a business process wherein trading partners use technology and a standard set of business processes for Internet-based collaboration on forecasts and plans for replenishing product. For the purpose of this study, We use CPFR as the reference to define business scenarios where trading partners (i.e., retailers, warehouses and manufacturers) have visibility into one another's critical demand, order forecasts and promotional forecasts using Web Services [7].

The need for a CPFR model arose from the fact that the supply chain inefficiencies were caused as a result of the lack of information sharing among trading partners. Also, there was a need for information to be available quickly to all trading partners so that timely action could be taken to meet unexpected events. For example, a spike in demand will register in time for everyone in the supply chain to effectively adjust and ensure proper inventory levels on store shelves. Costs for both trading partners are greatly reduced because problems can be anticipated and corrections made proactively. The capability for timely reaction is the gift from the Internet that has made meaningful collaborative relationships possible. CPFR provides a logical solution to resolving historical supply chain inefficiencies. The Internet has provided a means for more robust and instantaneous exchange of information between trading partners. Its use, in

conjunction with the industry-defined CPFR processes, has enabled a systematic method for the reduction of and timely reaction to supply- and demand-chain inconsistencies.

To put CPFR into perspective, consider the extent to which the retailer and manufacturer each know about how, when, and where the end-consumer is buying. Retailers have immediate firsthand knowledge of consumer buying behavior learned through everyday observations, supported by their interpretation of POS data. On the other hand, manufacturers aggregate consumer buying behavior based on inferences made from syndicated data (that is almost always inherently delayed), or retailers' order quantities. Collaborative processes efficiently combine the knowledge of each, which ultimately eliminate surprises at the manufacturer that lead to out-of-stocks at the retailer. As a result, increased sales, organizational streamlining and alignment, administrative and operational efficiency, improved cash flow, and improved return on assets performance ultimately occurs for both parties.

A predecessor to the CPFR system is the Vendor Managed Inventory (VMI) system. A VMI system can be defined as a streamlined approach to inventory and order fulfillment and is a system in which a vendor continuously and automatically replenishes a trading partner's inventory. True VMI occurs between a distributor and a manufacturer, with Electronic Data Interchange (EDI) being the crucial link between the two companies. Table 2.1 compares the CPFR system to the VMI system, highlighting the benefits of a CPFR system over VMI.

Table 2.1 *Comparing a CPFR system to a VMI System*

CPFR	**VMI**
Common goals	Divergent goals
A single demand forecast developed collaboratively	Multiple demand forecasts developed independently
Collaborative promotional planning & execution	Inconsistent promotion execution
A single, shared data source	Lack of data synchronization

Improved in-stock positions and service levels	Challenged in-stock positions at retail, in DC's and at supplier
Improved inventory management across entire supply chain	Low visibility to supply chain inventories
Optimized replenishment strategies with joint ownership	Sub-optimized replenishment capabilities
Process simplicity creates optimal framework for success	Process complications between retailer & supplier lead to operational issues

2.1 CPFR Process Model

VICS defines a 9-step CPFR process model for collaboration in achieving a common sales and order forecast. Figure 2.1 illustrates the CPFR process model, clearly identifying the activities in Planning, Forecasting and replenishment phases of the process.

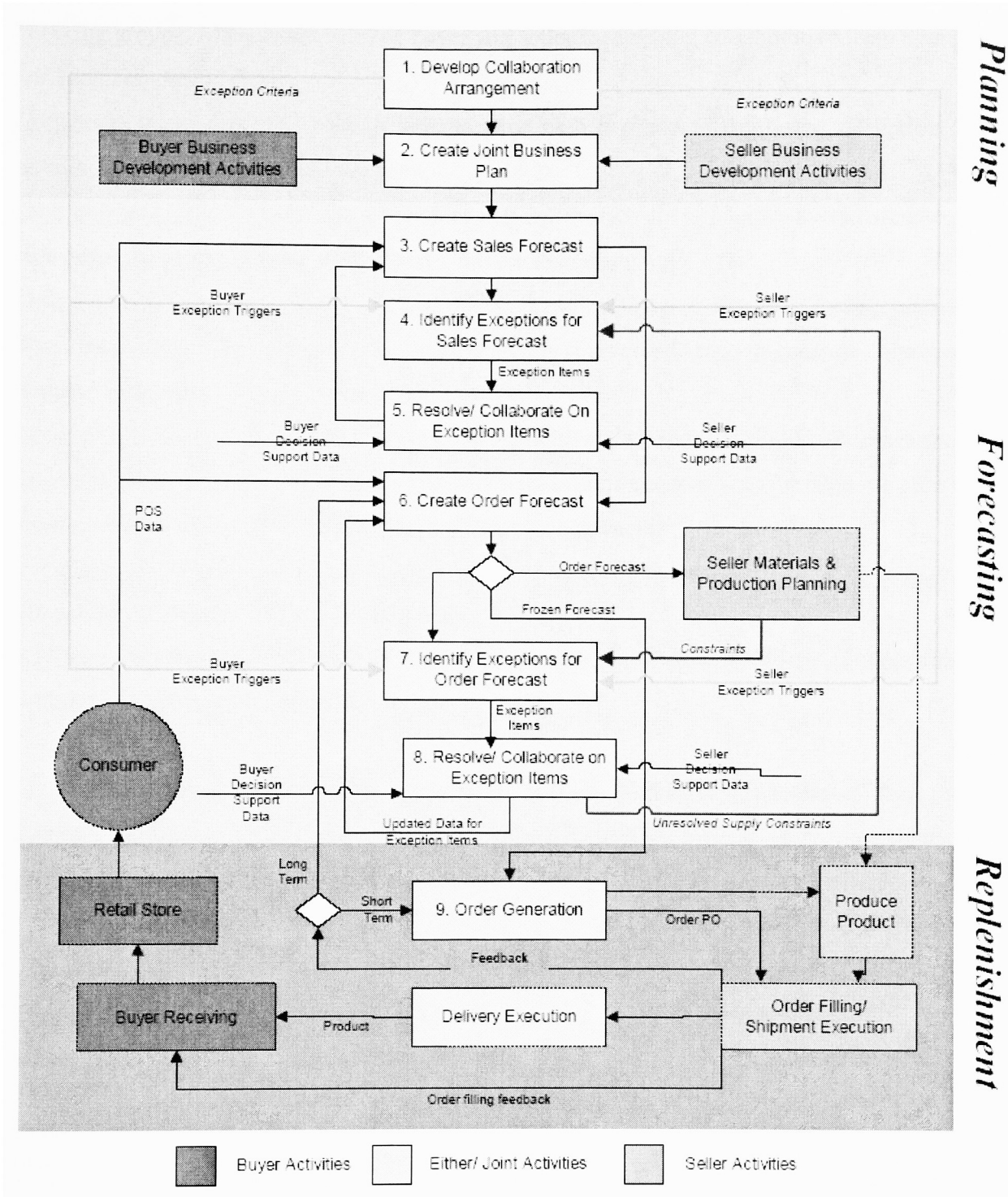

Figure 2.1. The 9-step CPFR process diagram.

The process diagram illustrates the key activities as described below.

Working groups: Manage the roles of buyer and seller by defining collaborative team members and their responsibilities.

Collaboration arrangement: Arrange collaboration such as product items, areas, exception criteria, and conditions of collaboration.

Joint business plan: Establish joint business plan collaboratively to identify event calendar (openings, closings, promotions, holidays, advertisements, etc.).

Demand collaboration: Integrate demand planning and exchange forecasts of trading partners. This function provides mechanism to collaboratively forecast and then to generate common demand forecasts.

Order collaboration: The demand forecast, on-hand inventory and shipping information are integrated to collaboratively generate and exchange order forecasts. This function includes the frozen mechanism of order forecasting and order management.

Exception list: Monitor and automatically trigger the exceptions and then give notices in order to allow trading partners to control exceptions.

KPI & report: Managers can analyze the key performances and download reports in accordance with each warehouse or each product item.

This process model also illustrates that data flows from each partner at each step, thereby getting consumed and produced at each end. Table 2.2 identifies the data consumed and data produced for each step in the CPFR process.

Table 2.2 *Input and Output Data at Each Step in CPFR Process*

Process Step	**Data Consumed**	**Data Produced**
Develop front end agreement	None: Manual Process	None: Manual Process
Create joint business plan	1. Buyer's corporate strategy 2. Seller's corporate strategy	Joint business plan
Create sales forecast	1. Joint business plan 2. POS data 3. Event 4. Sales forecast revisions	Sales forecast
Identify sales forecast exceptions	1. Sales forecast 2. Exceptions 3. Metrics	Identified exception items

	4. Events	
Collaborate on sales forecast exceptions	1. Buyer's secondary data for exception items 2. Identified exception items 3. Seller's secondary data for exception items	Sales forecast item revisions
Create order forecast	1. Order forecast revisions 2. POS data 3. Current inventory on hand 4. Inventory strategy/Seasonal Info 5. Sales forecast 6. Events 7. Product historical demand and shipments 8. Product availability data	Order forecast
Identify order forecast exceptions	1. Order forecast 2. Exceptions criteria and values 3. Events	Identified order forecast exception items
Collaborate on order forecast exceptions	1. Buyer's secondary data for exception items 2. Identified exception items 3. Seller's secondary data for exception items	Order forecast item revisions
Generate Order	1. Order forecast 2. Item management profile	Order

Researches were done on quantifying the benefits of using the CPFR model to achieve supply chain collaboration over the Internet. In the case study on implementation of CPFR at CatCo, a pet products vendor, CatCo collaborated with its retailers over the Internet using standard technology agreed upon by CatCo and its retailers. The benefits that were realized are listed below [23].

- A sales increase of 5-15% over the average
- In-stock inventory increased to over 98%
- Inventory levels lowered to between 25-50% across the supply chain

Although CatCo was able to collaborate over the Internet, the dependence on a particular technology limits the use of CPFR by all its business partners, as not every business partner was

willing to adopt the technology that CatCo was willing to use. There was a need for a vendor-independent and technology-neutral architecture for sharing information [23].

In the Case study on CPFR implementation between Johnson and Johnson (J and J) and Superdrug, J and J was able to reduce stock by 13% and increase the warehouse availability by 1.3%. J and J and Superdrug agreed upon a common technology and were restricted to using databases for accessing their data. The effort to integrate J and J with other retailers did not materialize as it proved costly to implement a system using a technology compliant with what J and J used. Restricting the data sources in CPFR created the additional burden of moving the data to databases for collaboration and of maintaining them [24].

This presents the need to have a vendor-independent and technology-neutral solution to share information over the Internet across the supply chain, and to aggregate data from different data sources such as databases, XML and Excel files. This thesis presents an architecture for CPFR where collaboration takes place independent of data sources and technology [24].

2.2 Key Performance Indicators in CPFR

Key performance indicators (KPI), also known as key success indicators (KSI), help an organization define and measure progress toward organizational goals. Once an organization has analyzed its mission, identified all its stakeholders, and defined its goals, it needs a way to measure progress toward those goals. Key performance indicators are those measurements.

Key performance indicators have three key features.

1) Key performance indicators reflect the organizational goals.
2) Key performance indicators must be quantifiable.
3) Key performance indicators must be key to organizational success.

Key performance indicators are quantifiable measurements, agreed to beforehand, that reflect the critical success factors of an organization. They will differ depending on the organization. A business may have as one of its key performance indicators the percentage of its

income that comes from return customers. A supply chain system may focus its key performance indicators on service levels and lead times of its partners.

Whatever key performance indicators are selected, they must reflect the organization's goals, they must be key to its success, and they must be quantifiable (measurable). Key performance indicators usually are long-term considerations. The definition of what they are and how they are measured does not change often. The goals for a particular key performance indicator may change as the organization's goals change, or as it gets closer to achieving a goal.

In the CPFR context, KPIs are used to measure the overall performance of the relationship between manufacturer and retailer. Not all of these KPIs are required by CPFR projects to measure the performances. Only the KPIs most relevant to the current relationship of trading partners should be selected in order to facilitate implementation of CPFR. Besides, the responsibilities for measuring each KPI should be assigned according to each partner's business processes and ability to measure the results. ECR (Efficient Consumer Response) Europe classified the key performance indicators (KPIs) of CPFR into the ten categories listed in Figure 2.3 [22].

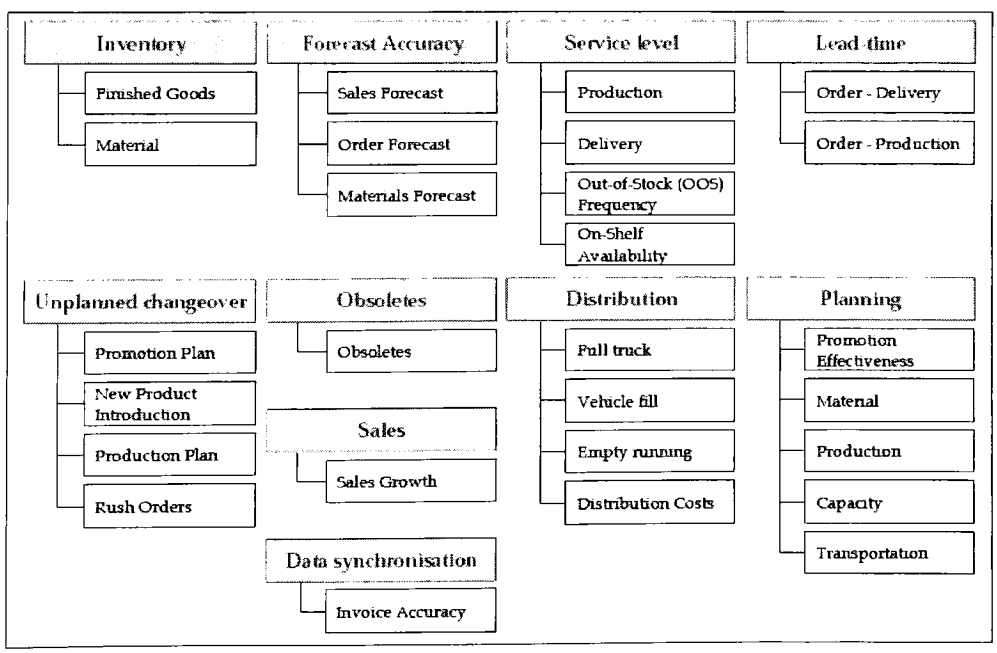

Figure 2.3. Key performance indicators for CPFR [22].

The agreement on KPIs by both trading partners is essential for successfully tracking the progress of the collaboration and in order to measure the benefits achieved through CPFR. Moreover, these definitions for KPIs leave enough scope to adjust them according to specific situations such as different forecast periods or frozen time periods.

The definitions and calculation formulas for a few of the KPIs used in CPFR are described in Table 2.2.

Table 2.2 *KPI Definitions and Calculating Formulas*

KPI	**Definition**	**Calculating formula**
Order forecast accuracy	Order forecast value in a defined time before actual order, divided by the actual order value	1-(|actual order value - Order Forecast value| / actual order value)*100%
Finished goods production lead-time	Number of days it takes from buyer delivering order to seller shipping goods	Seller ship goods date - Buyer delivery order date
On time delivery	Number of orders delivered on time divided by the total number of orders	(Number of orders delivered on time / the total number of orders)*100%
Order fill rate	Quantity of goods shipped on time divided by the total quantity of orders during a defined period of time	(Quantity of goods shipped on time / the total quantity of orders)*100%

2.3 Executive Dashboards

Executives and business-line managers often require real-time information to monitor operations and make decisions. Executive dashboards are highly customized
Software applications that provide the ultimate level of visibility into their operations using familiar, instantly understood visuals for key performance indicators (or KPIs). Unlike existing reporting systems or business "dashboard" solutions that use data that are vendor-dependent, the need today in an Internet-enabled world is for a reporting system to access data independent of vendor, i.e., from disparate data sources. Executive dashboards can be defined as tools to collect, unify, and display data from an array of systems, transforming it all into meaningful, actionable information [26].

Dashboards are cognitive tools that improve the "span of control" over a lot of business data. These tools help executives in visually identifying trends, patterns and anomalies, reasoning about what they see, and being guided toward effective decisions. As such, these tools need to leverage people's visual capabilities. With the prevalence of scorecards, dashboards and other visualization tools now widely available for business users to review their data, the issue of visual information design is more important than ever [26].

Since dashboards are used to manage the performance of a supply chain, a few key attributes are essential in any executive dashboard. A few key features of dashboards are listed below [26].

1) It must present information in the context of the user. A finance executive would have a view of the pricing part of the order whereas an operations executive would view the information on the order forecasts and Available To Promise (ATP) information.
2) It must be focused on productivity; i.e., a dashboard must provide a big picture of the organization's performance, and not a small subset of it.
3) It must provide an active and always-on interface; i.e., provide real-time up-to-the-second information.
4) It must be easier to use and easier to understand. A simple user interface improves productivity and assists in quick actions.
5) It must provide drill-down intelligence; i.e., users must be able to get multilevel visibility into the data. For example, order information available as quantities should be viewed yearly or drilled down to be viewed monthly.
6) It must provide role-based security in cases where data is restricted to only some users.

As CPFR gained significance, there was a need for enterprises to aggregate data from different data sources. The aggregated data when available in real time would enable executives to make quick decisions and would enable them to better manage the supply chain. With the emergence of the Internet as the mode of communication between business partners, several technologies have emerged for enabling data and application integration. The technologies that enable supply chain collaboration using the Internet are described in the next chapter.

3. XML and Emerging XML Technologies

XML stands for Extensible Markup Language and is a universally agreed-upon markup language primarily used for information exchange. A good example of a markup language is the Hyper Text Markup Language (HTML). The beauty of XML lies in the fact that it is extensible. Simply put, XML is a set of predefined rules (syntactical framework) that we need to follow when structuring our data. For a long time, programmers and application vendors have built applications and systems to be deployed in an enterprise for processing data that can be interpreted by the enterprise systems—essentially, data structured in a proprietary fashion. But as information exchange between applications and systems across enterprises became prevalent, it became very difficult to exchange data because the systems were never designed to accept data from external, unknown systems. XML provides a standard and common data structure for sharing data between disparate systems. Additionally, XML has built-in data validation, which guarantees that the structure of the data that is received is valid [29].

Extensible Markup Language (XML) is an extensible, portable, and structured text format. XML is playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. XML was derived from SGML, which was a complex language for defining other markup languages [29].

The XML initiative consists of bunch of related standards. Apart from the core XML standard, it includes XSL—Extensible Stylesheet Language, which is used to transform XML data into a customizable presentation. XLink and XQuery provide a way to provide flexible query facilities to extract data from real and virtual XML documents on the Web. XPath and XPointer are languages for addressing parts of an XML document [29].

3.1 Components of XML [29]

When working with XML, we think of creating XML documents and consuming XML documents. The creation process involves using editors and tools to create XML documents. On the other hand, consuming XML documents involves parsing the XML documents and extracting the useful data. Creating XML documents is a two-step process, which involves:

1) Defining the grammar and restrictions over data for the XML document

2) Creating the XML document itself

This document can be validated against the grammar. The DTD and Schema are used to describe the grammar and restriction over data in the XML document. The basic components of XML are discussed below.

Elements [29]:

Elements are tags, just like the ones used in HTML, and have values. Further, elements are structured as a tree. Hence, we have elements organized in a hierarchical fashion with a base element (parents element) and child elements; child elements themselves further can have more child elements, and so on. In the preceding example, <employee> is the root element and has <shift> as its child element; further down, <phone> is the child of <shift>.

Elements have certain characteristics. Some of these characteristics are:

Elements can contain data, such as the <number> element in the example. On the flip side, elements may not contain data but just attributes, such as the <shift> element. Alternatively, elements may have both attributes and data, and may also contain child elements, like the <phone> element in the example. There are many more features and rules associated with elements, such as what valid names an element tag can have, the requirement for elements to be properly nested, and so on.

Attributes [29]:

Attributes help us to give more meaning and to describe our element more efficiently and clearly. In the preceding example, the <shift> element has an attribute "id" with values "counter" and "help_desk." With the use of such attributes, we can easily know that an employee can be working at a counter or help desk. This helps make the data in the XML document self-describing. It is important to note that the core purpose of attributes is to provide more information about the element, so they should not be used to contain the data itself. Just as with elements, attributes have many rules associated with them.

3.2 DTD and XML Schema [29]

DTD and Schema are used to specify the structure of instance documents and the datatype of each element/attribute. DTDs used today in XML originated from the parent SGML specification. Because SGML was designed for a more document-centric model, it did not require the use of complex datatyping definitions. The XML Schema specification improves greatly upon the DTD content model by providing rich datatyping capabilities for elements and attributes as well as providing OO design principles.

3.2.1 Document Type Definition (DTD) [29]

Just as when we start coding in any programming language we need to know the language specification, in the same way DTD is a specification, which has to be followed while creating an XML document. Also, just as one of the tasks of the compiler for any programming language is to see if the specification was followed, similarly there are parsers that use the DTD to check the validity of the XML document.

A DTD helps us to define the structure of our XML document. It provides a strict framework and rules to be followed when creating XML documents. In addition, DTD can be used to check the validity and integrity of the data contained in an XML document. A few salient features of DTD are listed below:

DTD is used to specify valid elements and attributes that can be used in the XML document. With a DTD, we can define a tree hierarchy of the elements. Sequential organization of a collection of child elements that can exist in an XML document can also be defined by using a DTD. DTD can be used directly inside the XML source or can exist outside an XML document with a link specified in the XML document to that DTD.

Basically, DTD consists of these items:

1) DTD Element: Metadata about an element. It specifies what kind of data an element will have, the number of occurrences of each element, relationships between elements and so on.
2) DTD Attributes: Specify various rules and specifications associated with the data
3) DTD Entities: Used to reference an external file or provide shortcuts to common text

Example: <!ELEMENT employee (shift+, home-address, hobbies*)>
employee can have one or more shift a day and should have one home-address and can have zero or more hobbies.

Example: <!ATTLIST shift id CDATA #REQUIRED>
shift should have an id attribute.

In short, DTD is used to define a document structure by specifying the details regarding all the elements that are to be used and hence can be used to check the validity of an XML document that is supposed to follow the rules laid down by this DTD.

3.2.2 XML Schema [29]

XML Schema is a more advanced version of DTD. XML Schema was approved as a W3C Recommendation in May, 2001 and is now being widely used for structuring XML documents for e-commerce and Web Services applications. The two major goals that the W3C XML Schema working group focused on during the design of the XML Schema standard were:

1) Expressing Object Oriented (OO) design principles found in common OO programming languages into the specification
2) Providing rich datatype support similar to the datatypes available in most relational database systems

XML Schema provides a means of creating a set of rules that can be used to identify document rules governing the validity of the XML documents that we create. It also provides a

means of defining the structure, content, and semantics of XML documents that can be shared between different types of computers and documents.

The most common features of XML Schema are [29]:

1) Syntax is very similar to XML. This means we can edit our Schema by using any XML editor. We not only specify basic data types like string, integer, long, float, and so forth, but also can define our own custom data types. Example:

<xs:element name="name" type="xs:string"/>

The new types we can define are *simple* and *complex*. Complex types may contain other elements and/or attributes, whereas simple types do not contain other elements or attributes. Instead, they contain only simple text data.

2) XML Schema provides Content-Based Validation (the order in which the child elements are nested) and also provides Data Type validations, with lots of functionality and validation checks provided for simple and complex types.

For example, we can define a simple type with a 'year' range between 2000 and 2100 as follows:

```
<xsd:simpleType name="year">
 <xsd:restriction base="xsd:integer">
  <xsd:minInclusive value="2000"/>
  <xsd:maxInclusive value="2100"/>
 </xsd:restriction>
</xsd:simpleType>
```

Similarly, Complex types can describe the restrictions on the sequence in which the child elements should appear. Example:

```
<xsd:complexType name="Product">
 <xsd:sequence>
  <xsd:element name="Name" type="xsd:string"/>
  <xsd:element name="Quantity" type="xsd: integer "/>
  <xsd:element name="Price" type=" xsd:number "/>
```

```
 </xsd:sequence>
</xsd:complexType>
```

3) XML Schema provides us with the ability to extend other documents, which is nothing but inheritance in Object-Oriented terms. This means we can reuse and refine other Schema definitions.

4) Support for Namespace (using URI) is also provided by XML Schema. It provides each element a unique identifier, which avoids element name conflicts that may occur due to many reasons—say, when two documents are merged, and both documents have "name" fields but have different meanings for them. For example, "name" can be a products's name in one document, but it can be a raw material's name in another document. In short, the Schema helps in distinguishing duplicate elements and attributes.

Here, the 'id' is a prefix and namespace is 'http://somesite.com/schema'.
After the namespace is defined, we can use the prefix on all the elements to uniquely identify it.

```
<myElement xmlns:id='http://somesite.com/schema'>
 <id:name>myName</id:name>
</myElement>
```

5) Last but not least, XML Schema is easily extendible to incorporate more features in the future.
XML Schema offers many advantages over DTD. The advantages of XML Schema over DTD are listed below.

1) Enhanced data types:

XML Schema supports over 44 datatypes versus 10 supported by DTD. we can create our own data type in XML Schema. For example: "This is a new type based on the string type, and elements of this type must follow this pattern: ddd-dddd-ddd, where 'd' represents a digit."

2) Written in the same syntax as instance documents:

DTD requires us to remember another syntax than the one used to write XML documents. Remembering an extra set of syntax represents overhead and is error-prone.

3) Object-oriented:

Schemas can extend or restrict a type (derive new type definitions on the basis of old ones). Schemas can specify element content as being unique (keys on content) and can specify uniqueness within a region. They can also define multiple elements with the same name but different content, by using namespaces. Lack of namespaces was a major drawback in DTD. We can think of namespaces in XML like namespaces in C++. A simple analogy of DTD vs. XML Schema namespace usage is the use of global and local variables in programming languages. A local variable's name is unique within its scope, whereas a global variable has to be unique across functions. Similarly, with an XML Schema namespace we have freedom to define datatypes without worrying about name collisions.

4) Well formedness and Validity:

Well formedness of an XML document (also known as an instance document) refers to the characteristic of a document adhering to the XML rule of well-formedness. As we would recall, XML has a stringent set of rules, unlike HTML, such as closing all the tags, no nested tags, and so forth.

One quick way of checking well formedness of an XML document is opening it in a browser window. Both Internet Explorer and Netscape provide automatic well-formedness checking and display any errors in the browser. Validity of an instance document implies that the document conforms to the specified Schema or DTD file mentioned in the XML document.

3.3 Web Services

A Web Service is an application that can be called over the Web using standards such as SOAP over HTTP. Web Services combine the best aspects of component-based development and the Web. Like components, Web Services represent black-box functionality that can be reused without worrying about how the service is implemented. Unlike current component

technologies, Web Services are accessed via ubiquitous Web protocols (ex: HTTP) and data formats (ex: XML). Web Services is a term that is being used to define a set of technologies that exposes business functionality over the Web as a set of automated interfaces. These automated interfaces allow businesses to discover and bind to interfaces at run-time, supposedly minimizing the amount of static preparation that is needed by other integration technologies [23].

To summarize: a Web Service . . .

- Is a programmable application, accessible as a component via standard Web protocols
- Uses standard Web protocols like HTTP, XML and SOAP
- Works through existing proxies and firewalls
- Can take advantage of HTTP authentication
- Features encryption with SSL
- Is easily incorporated with existing XML messaging solutions
- Takes advantage of XML messaging Schemas and allows easy transition from XML RPC solutions
- Avoids conflict between proprietary component-based solutions like CORBA and COM
- Combines the best aspects of component-based development and the Web, and
- Is available to a variety of clients (platform independent) [23]

3.3.1 Architecture of Web Services

Client/server applications in use in the 80s and early 90s were primarily data-centric in nature. The focus was more on retrieval and display of data than on the business rules of the application. Web application architectures changed all this. Because Web applications are based on a distributed architecture, Web applications have moved away from the data-centric nature of client/server applications and tended more to a service-oriented architecture. The basic distributed technologies from Remote Procedure Calls (RPC), to Remote Method Invocation (RMI), to Common Object Request Broker Architecture (CORBA) are inherently service-oriented. Web applications have evolved from simple applications accessed and deployed in an Intranet to applications of different businesses/organizations "talking" to each other and

exchanging data. The next step for Web applications is to provide business services that can be accessed by other applications of different businesses/organizations using the service-oriented architecture of Web Services. Moreover, the advantages of re-use in distributed Web applications have resulted in applications being designed to provide more business-related services [23].

When we talk about service-oriented architecture for Web Services applications, quite a few things come up. The application providing a service and the client application using the service talk to each other in a common language. Without both applications talking in the same language and exchanging information, the entire architecture will turn into one more challenge to distributed enterprise applications. Next, a service-providing application and a client application using the service need some way to locate each other before they start talking with each other. This is especially true for distributed applications, where applications have no knowledge of each other's location [23].

Hence, we can conclude that a basic service-oriented architecture for Web Services has:

- a standard way of communication
- a uniform data representation and exchange mechanism
- a standard meta language to describe the services offered
- a mechanism to register and locate Web Services-based applications

Figure 3.1 displays the building blocks of a Web Service. A Client consumes a provider Web Service using a standard communication channel.

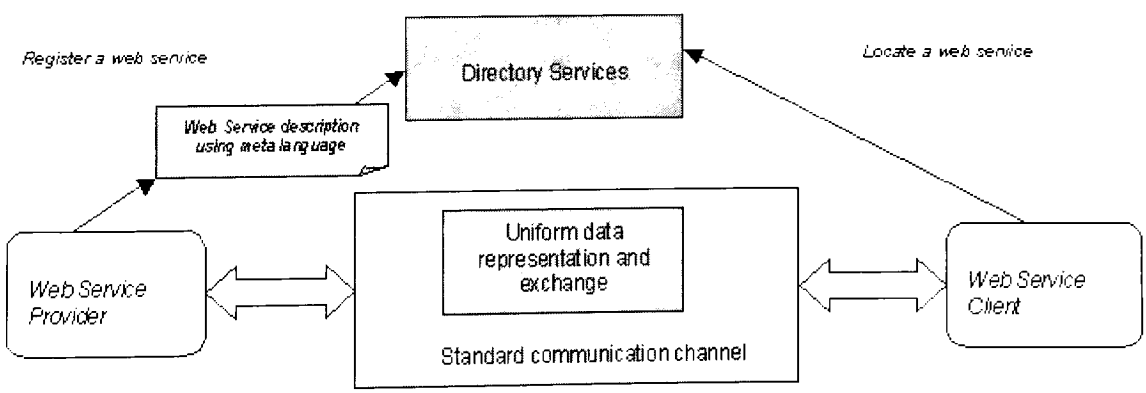

Figure 3.1. Architecture of Web Services. [23]

A brief overview of the different technologies involved in Web Services development is discussed below.

3.3.2 Technologies Used in Web Services [23]

The mapping between the different layers of the Web Services architecture and the technologies used are described in table 3.1.

Table 3.1 *Web Service Layers and the Technology Used* [23]

Layer	**Technology**	**Description**
Uniform data representation and exchange	XML	Extended Markup Language (XML) is a meta language that has a well-defined syntax and semantics. The "self describing" syntax and semantics features of XML make it a simple, yet powerful, mechanism for capturing and exchanging data between different applications. XML is a tried and tested way for exchanging data and has been extensively used in B2B applications. Hence, XML is used in the Web Services architecture as the format for transferring information/data between a Web Services provider application and a Web Services client application. The data embedded in an XML file is generated as well as processed using a powerful set of XML parser APIs, viz. Simple API for XML (SAX) and Document Object Model (DOM) parser API.
Standard communication channel	SOAP	The Simple Object Access Protocol (SOAP) is the channel used for communication between a Web Services provider application and a client application. The simplicity of SOAP is that it does not define any new transport protocol; instead, it re-uses the Hyper Text Transfer Protocol (HTTP) for transporting data as messages. This use of HTTP as the underlying protocol ensures that Web Services provider applications and client applications can communicate using the Internet as the backbone. It is the use of SOAP that multiplies the capabilities of Web Services and makes it all the more exciting!
Standard meta	WSDL	Web Services provider applications advertise the different

language to describe the services offered		services they provide using a standard meta language called the Web Services Description Language (WSDL). Interestingly, WSDL is based on XML and uses a special set of tags to describe a Web Service, services provided, where to locate it, and so forth. Client applications obtain information about a Web Service prior to accessing and using a Web Service of a Web Service provider.
Registering and locating Web Services	UDDI	The "yellow pages" or directory of Web Services is the Universal Description Discovery and Integration (UDDI). Web Services application providers are listed in a registry of service providers using UDDI. Similarly, client applications locate Web Services application providers using UDDI. As in the case of WSDL, UDDI also is based on XML.

A quick overview of the general steps involved in building a Web Services application is discussed below.

3.3.3 Basic Steps in Web Services Application Development

Building applications based on Web Services is similar to building applications based on distributed technologies such as RPC, RMI, CORBA, or EJB because the basic building blocks are similar. The basic building blocks in a Web Services application for a Client or Service User and a Service or a Service User are illustrated in Figure 3.2 below.

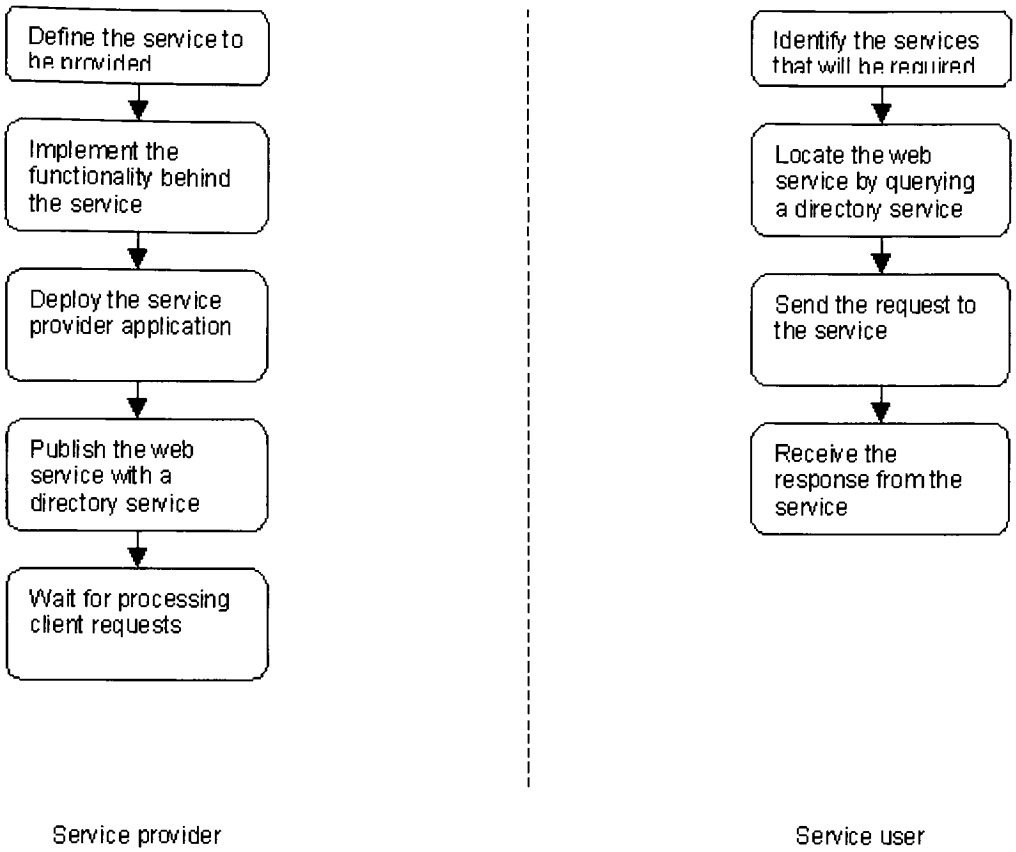

Figure 3.2. Steps for building a Web Services application.

These steps will be the same irrespective of which major technology/programming language we use to implement the Web Service.

The following are the key benefits of using a Web Service [23].

1) Software as a service as opposed to packaged products: Web Services can be delivered and paid for as streams of services, and they allow ubiquitous access from any platform. Web Services allow for encapsulation. Components can be isolated such that only the business-level services are exposed. This results in decoupling between components, and thus more stable and flexible systems.

2) Dynamic business interoperability: New business partnerships can be constructed dynamically and automatically since Web Services ensure complete interoperability between systems.

3) Accessibility: Business services can be completely decentralized and distributed over the Internet and accessed by a wide variety of communications devices.

4) Efficiencies: Businesses can be released from the burden of complex, slow and expensive software development and focus instead on value-added and mission-critical tasks. Web Services constructed from applications meant for internal use can be easily exposed for external use without changing code. Incremental development using Web Services is natural and easy, and since Web Services are declared and implemented in a human-readable format there is easier bug tracking and fixing. The overall result is risk reduction and more efficient deployability.

5) Universally agreed specifications: Web Services are based on universally agreed-upon specifications for structured data exchange, messaging, discovery of services, interface description, and business process orchestration.

6) Legacy integration: Web Services offer greater agility and flexibility from increased integration between legacy systems.

7) New Market Opportunities: There will be greater feasibility to the dynamic enterprise and dynamic value chain businesses.

3.4 XQuery

XML has rapidly gained popularity as a formatting language for information, finding constituencies in both the document-centric and data-centric worlds. The explosive growth of XML-based standards bears testimony to XML's interest to many different technical communities. Applications now use XML for both transient messages, such as SOAP messages, and as persistent storage, such as in XML databases or content management systems. An XML-based Web, as opposed to an HTML-based Web, no longer sounds like fantasy [6].

As the volume of information stored in XML grows, it becomes correspondingly more important to be able to access information in XML documents efficiently and effectively. To do that, we need an expressive query language so we can specify precisely what information we want to retrieve or update in an XML data source. XQuery intends to be that language.

XQuery is a query language specification under development by the World Wide Web Consortium (W3C) that's designed to query collections of XML data—not just XML files, but anything that can appear as XML, including relational databases.

XQuery provides the mechanism to efficiently and easily extract information from Native XML Databases (NXD) and relational data as well. With XQuery, we can view RDBMS tables as just another XML data source. XQuery makes possible the exciting possibility of a single query that combines an incoming purchase order in native XML format, an archive of catalog data also in native XML format, and an inventory system held in a relational database.

XQuery isn't limited to only the server, either. It's already showing up in products such as Apple's Sherlock as a mechanism for querying XML data feeds. Sherlock actually combines XQuery with JavaScript to produce an easy way to script and query XML content.

3.4.1 Brief History of XQuery

XQuery started its life as Quilt, which in turn was influenced by a number of other query languages, including XPath, XQL, XML-QL, SQL, OQL, Lorel, and YATL. The W3C XML Query Working Group was created in 1999 and has since produced specifications defining XQuery syntax and semantics. Many of the original contributors to the predecessor languages eventually became members of the Working Group. In 2000, the Working Group proposed several major working drafts, including requirements for XQuery, data model, use cases, and query algebra.

In June 2001, the latest working draft of the XQuery 1.0 specification was released by the W3C XML Query Working Group in collaboration with the W3C XSL Working Group. To support many powerful features of XQuery, several significant enhancements were proposed for XPath 2.0. While XPath has limitations as a query language for XML, it does provide foundation capabilities relied on by XQuery. The proposed working draft for XQuery 1.0 and XPath 2.0 Data Model was released in June 2001. The XML Query Formal Semantics specification (XQFS) replaced the XML Query Algebra spec. The latest addition to the list of specifications under the working group includes an XML syntax of XQuery, called XQueryX (no public draft) and a Functions and Operators document, describing the functions and operators on XML

Schema datatypes defined in XML Schema—Part 2. All the specifications produced by the W3C Query working group are works in progress at this time, and as such are subject to change.

3.4.2 XPATH

XQuery makes heavy use of XPath, an expression language used to select portions of an XML document. In fact, XQuery 1.0 and XPath 2.0 are under development by the same W3C working group, and their specifications are intertwined. XPath expressions behave in some ways like regular expressions, except they operate on XML nodes instead of characters. Both XPath and regular expressions can look somewhat cryptic, with lots of slashes and brackets, but both are incredibly powerful and charmingly elegant.

Historically, XPath has been used as part of Extensible Style Language Transformations (XSLT). Document Object Model (DOM) Level 3 and JDOM also support XPath as a way to select portions of a document without manual searching. In XQuery, XPath expressions are a type of simple query, although they are more commonly part of larger queries.

3.4.3 XQuery Expressions

While XPath alone is useful for simple data extraction, XQuery builds on XPath with FLWOR expressions.

3.4.3.1 FLWOR Expressions

FLWOR (pronounced "flower") expressions are the building blocks of XQuery. The name comes from the For, Let, Where, Order by, and Return keywords that make up the expression. The "For" clause provides a mechanism for iteration, and the "Let" clause allows variable assignments. The "For" and "Let" clauses specify a sequence of tuples (a tuple is an ordered set of values).

These tuples can then be filtered with a "Where" clause and ordered using an "Order by" clause. The "Return" clause at the end of an expression indicates what should be returned. The "Return" clause is evaluated once for every tuple surviving the "Where" clause and ordered according to

the "Order by" clause. The return value of the FLWOR expression is the ordered sequence of content generated by the "Return" clause as it evaluates each tuple.

FLWOR expressions allow the building of arbitrarily complex XML results and provide a mechanism to utilize multiple documents in a single query.

The "Order by" clause uses an XPath expression to select the price elements and uses a built-in function "min()" to calculate the minimum value. It is important to note that the XQuery comment is surrounded by {-- and --} tags.

3.4.3.2 Conditional Expressions

Like most languages, XQuery provides the conditional expression "if/then/else." Unlike most languages, the "else" clause is mandatory. That's because every expression in XQuery has to return a value. There's no support for simple statements. For example, we learn to return () from the "else" clause to indicate the empty sequence. In a procedural language, we just wouldn't have an "else" clause.

To experiment with an if/then/else clause, I have a query to retrieve all books and their authors, but after the first two authors for a book, I want to return any additional authors as <et-al/>:

```
for $b in document("books.xml")/bib/book
return
 if (count($b/author) <= 2) then $b
 else <book> { $b/@*, $b/title, $b/author[position() <= 2], <et-al/>,
$b/publisher, $b/price } </book>
```

This query reads book data from a "books.xml" URI, probably a local file. For each <book>, if the author count is <= 2, then I return the <book> directly. Otherwise I construct a new <book> element containing all the original data, except that I include only the first two authors, and after that, I append an <et-al/> element. I use a special function, position(), in the predicate here to return only the first two authors. I also use the cryptic $b/@* XPath expression, which refers to all the attributes on $b. Placing the attributes here at the head of the content sequence for <book> attaches all attributes to the new <book> element.

These XQuery expressions enable XQuery to efficiently process XML data and aggregate them. We will now use XQuery to implement several CPFR scenarios. The scenarios use data from disparate data sources.

4. CPFR Scenarios

Several scenarios for a supply chain in a CPFR model are developed to prove the concept of data integration via XQuery. The primary data sources are Web Services and their equivalent XML files, SQL server databases and CSV/Excel files. Three of the scenarios (1a, 2a and 3a) consolidate data into key performance indicators while the other scenarios (1b, 2b and 3b) are generating exceptions. The information from the different data sources is made available as XML data. XQuery is the language that is used to perform the data aggregation on the XML data. The XML engine used to convert the data sources to XML and to process the XQuery code is "LIQUID DATA" by BEA systems. The evaluation version of Liquid Data provides the functionality to configure the data sources and to program and execute XQuery. Appendix 1 contains the XQuery code, the XML Schemas that were developed to define the XML structure and the output XML for each scenario. All results are stored in an SQL database and/or presented on a dashboard (GUI interface developed in Visual FoxPro) for appropriate business decision making.

4.1 Scenario 1a – Calculating Sales Forecast Accuracy KPI

In this scenario, the manufacturer is interested in verifying the sales forecast accuracy for the last eight weekly periods. The Point of Sale (POS) data from the two retailers are consolidated and contrasted with the manufacturer's sales forecasts for sales forecast accuracy. The sales forecast accuracy is a key performance indicator (KPI) that can be used to refine future forecasts.

4.1.1 Data Sources Used

The data sources for this scenario are as follows:

1) An XML file, "**getPOS**," which holds the POS information for eight weekly periods, is considered as an equivalent to a **getPOS webservice** response from each retailer.

2) A **SQL server database** of the Manufacturer, "**CPFR-Manufacturer**," contains the sales forecast information; the "salesforecast" table contains the sales forecast data required for this scenario.

4.1.2 Processing Done

An XQuery program is written to compare the sales information available in the XML file for each retailer with the sales forecast data in the manufacturer's database for that retailer to calculate the sales forecast accuracy for each time period. This key performance indicator is calculated using the formula mentioned below.

Sales forecast accuracy in %

= 1 - [(actual POS quantity - sales forecast quantity)/actual POS Quantity]*100

4.1.3 Results Obtained

The output from the XQuery is XML data which can be made available as a Web Service. This Web Service outputs an XML response that pertains to an XML Schema for the result. The XML Schema contains the following elements:

1) Seller ID
2) Buyer ID
3) Start date
4) End date
5) Actual Sales
6) Sales Forecast
7) Sales Forecast Accuracy

Figure 4.1 depicts the information flow in this scenario.

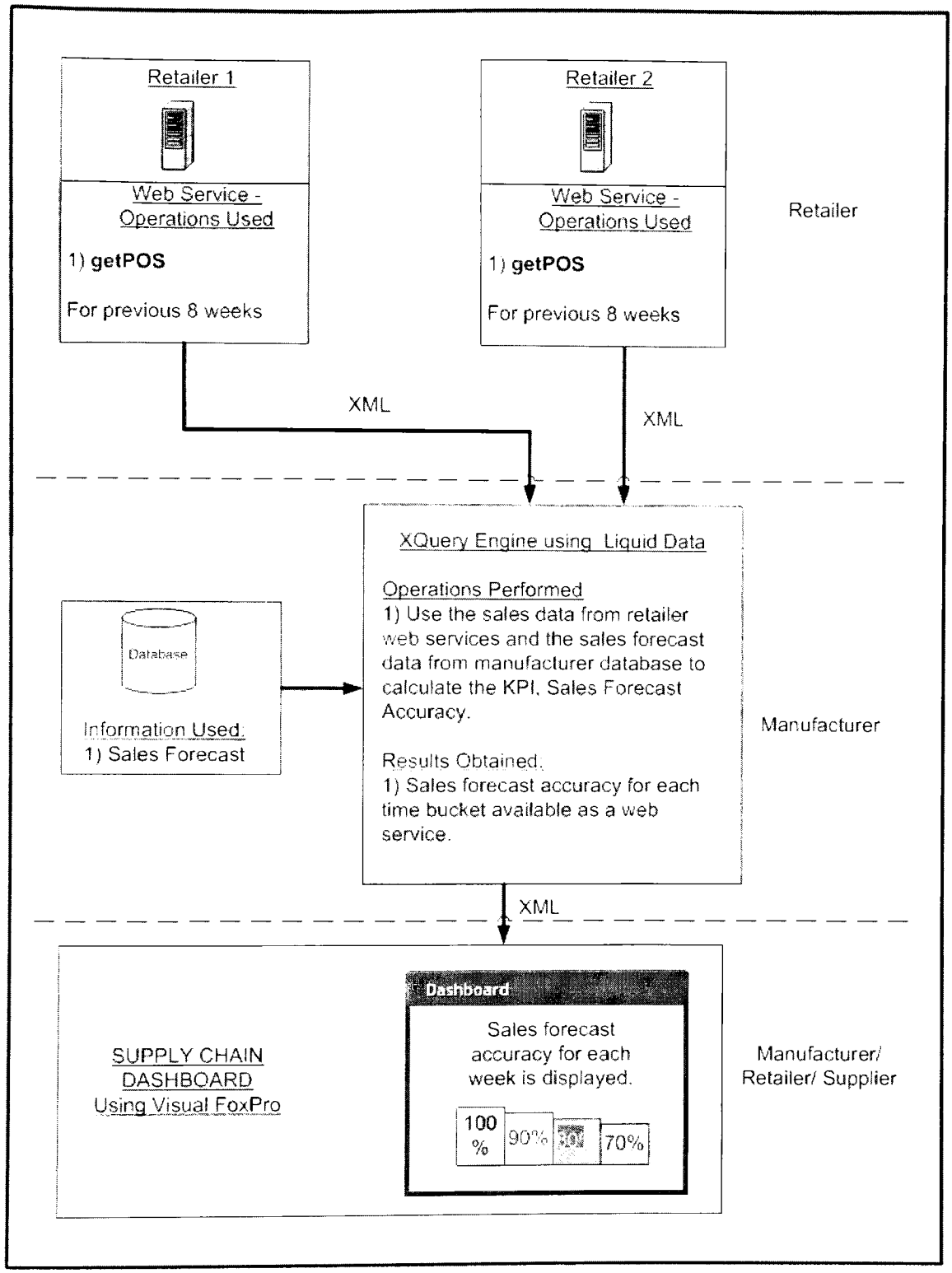

Figure 4.1. Scenario 1a – Information flow for calculating the Sales forecast accuracy.

4.2 Scenario 1b – Identifying Sales Forecast Exceptions

In this scenario, the manufacturer is interested in checking the variation in the sales forecast for the next eight weekly periods. The variations (more than 10%) in the future sales forecasts for the two retailers and the manufacturer are identified as **exceptions** for further negotiation in arriving at a final sales forecast.

4.2.1 Data Sources Used

The data sources for this scenario are as follows:

1) An XML file, "salesforecast," which holds the information about the sales forecast for the next eight time buckets, is considered as an equivalent to a **getSalesforecast webservice** response from each retailer.

2) A SQL server database of the Manufacturer, "CPFR-Manufacturer," contains the Sales forecast information; the "salesforecast" table contains the sales forecast data required for this scenario.

4.2.2 Processing Done

An XQuery program is written to compare the sales information available in the XML file for each retailer with the data in the manufacturer's database for that retailer. If the sales forecast variation is more than 10%, the system flags the output for that time bucket to contain an exception. An exception can be defined as a variation in the sales forecasts that is too large, such that the business partners must collaborate and resolve to arrive at a common sales forecast.

4.2.3 Results Obtained

The output from the XQuery is XML data which can be made available as a Web Service. This Web Service outputs an XML response that pertains to an XML Schema for the result. The XML Schema contains the following elements:

1) Seller ID
2) Buyer ID
3) Seller forecast quantity
4) Buyer forecast quantity
5) Has Exception
6) Exception Type ID
7) Difference

Figure 4.2 depicts the information flow from the various data sources to the XQuery engine in this scenario. The output is available as a Web Service.

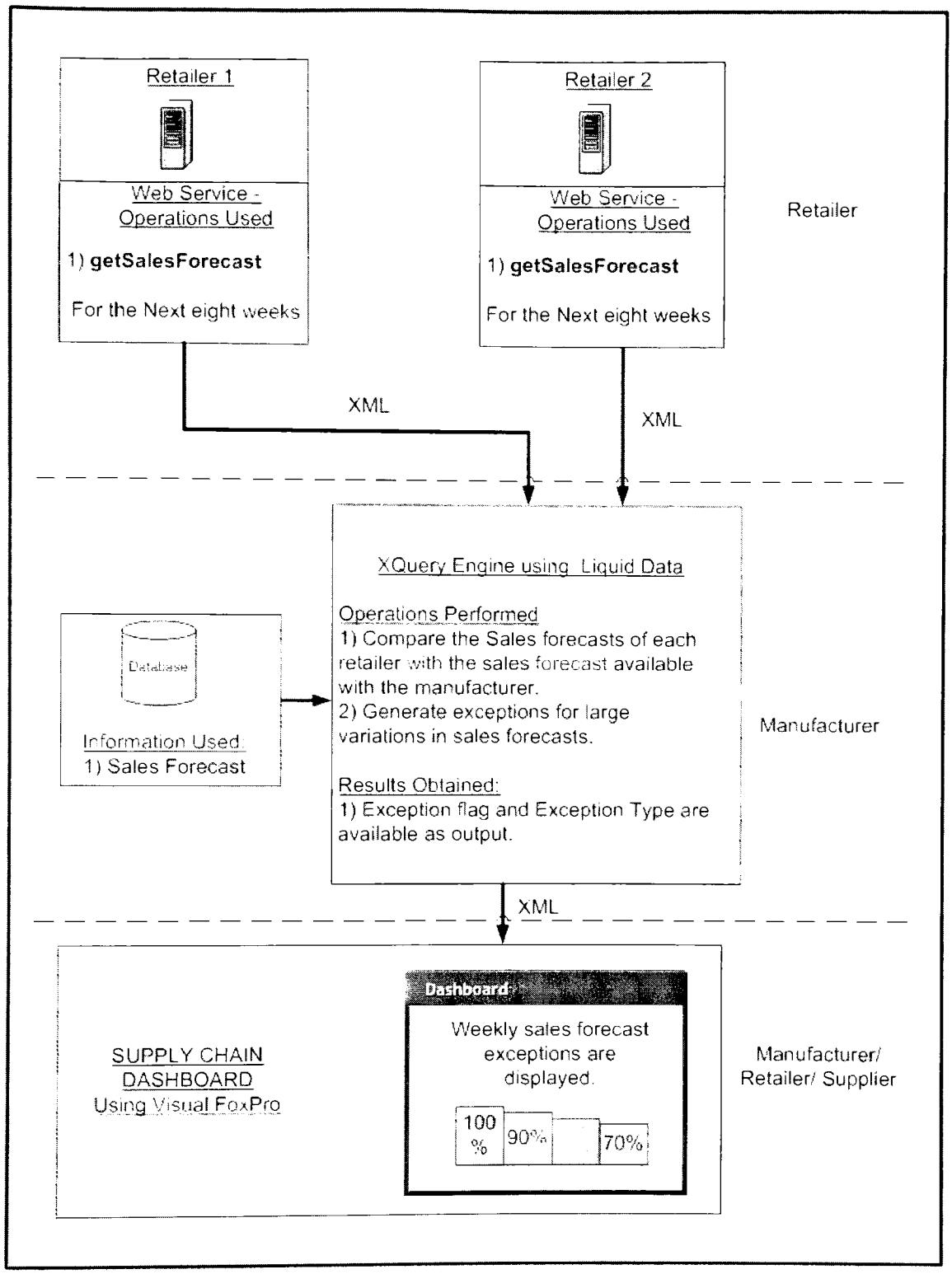

Figure 4.2. Scenario 1b – Information flow for identifying exceptions in Sales forecast.

4.3 Scenario 2a – Calculating Out of Stock Frequency KPI

In this scenario, the manufacturer is interested in verifying the promotion effectiveness for the last eight weekly periods. Promotion effectiveness refers to the valuation of the impact of a promotion once it has been done. The performance metrics to evaluate the promotion effectiveness are the out of stock (OOS) frequency and promotion forecast accuracy. The out of stock frequency refers to the number of times an item goes out of stock during the promotion period. Promotion forecast accuracy indicates if the promotions during the last eight weekly periods met the retailers' expectations. The point of sale (POS) data is compared with promotion forecast data for each retailer to generate the promotion forecast accuracy. The out of stock frequency and promotion forecast accuracy are key performance indicators (KPI) that can be used to plan forecasts on promotions in the future.

4.3.1 Data Sources Used

The data sources for this scenario are as follows:

1) An XML file, "**getPOS**," which holds the POS information for eight weekly periods, is considered as an equivalent to a **getPOS webservice** response from each retailer.
2) An XML file, "**getOnHandInventory**," which holds the on-hand inventory information for eight weekly periods, is considered as an equivalent to a **getOnHandInventory webservice** response from each retailer.
3) A **SQL server database** of the Manufacturer, "**CPFR-Manufacturer**," contains the promotion information for the previous eight weeks; the "promotionforecast" table contains the promotion forecast data required for this scenario.

4.3.2 Processing Done

An XQuery program is written to aggregate data from the data sources defined and to calculate the key performance indicators, out of stock frequency and promotion forecast accuracy.

The OOS frequency is calculated by obtaining the on-hand inventory quantities for each item during the weekly periods when there were promotions. If the on-hand inventory is found to be zero, it means that the item went out of stock for that period. The OOS frequency can then be calculated as,

OOS frequency in %

= [number of periods during promotions when the item went OOS - number of periods when promotions were held]*100

The promotion forecast accuracy is calculated by comparing the promotion forecast for the previous eight weekly periods available in the manufacturer's database with the actual POS information during the promotional periods. The promotion forecast accuracy is then calculated as,

promotion forecast accuracy in %

= 1 - [(actual POS quantity during the promotion period - promotion forecast quantity)/actual POS Quantity]*100

4.3.3 Results Obtained

The output form the XQuery is an XML data which can be made available as a Web Service. This Web Service outputs an XML response that pertains to an XML Schema for the result. The XML Schema contains the following elements:

1) Seller ID
2) Buyer ID
3) Start date
4) End date
5) Item ID
6) Out of stock frequency
7) Actual Sales
8) Promotion forecast quantity
9) Promotion Forecast Accuracy

Figure 4.3 depicts the information flow in this scenario.

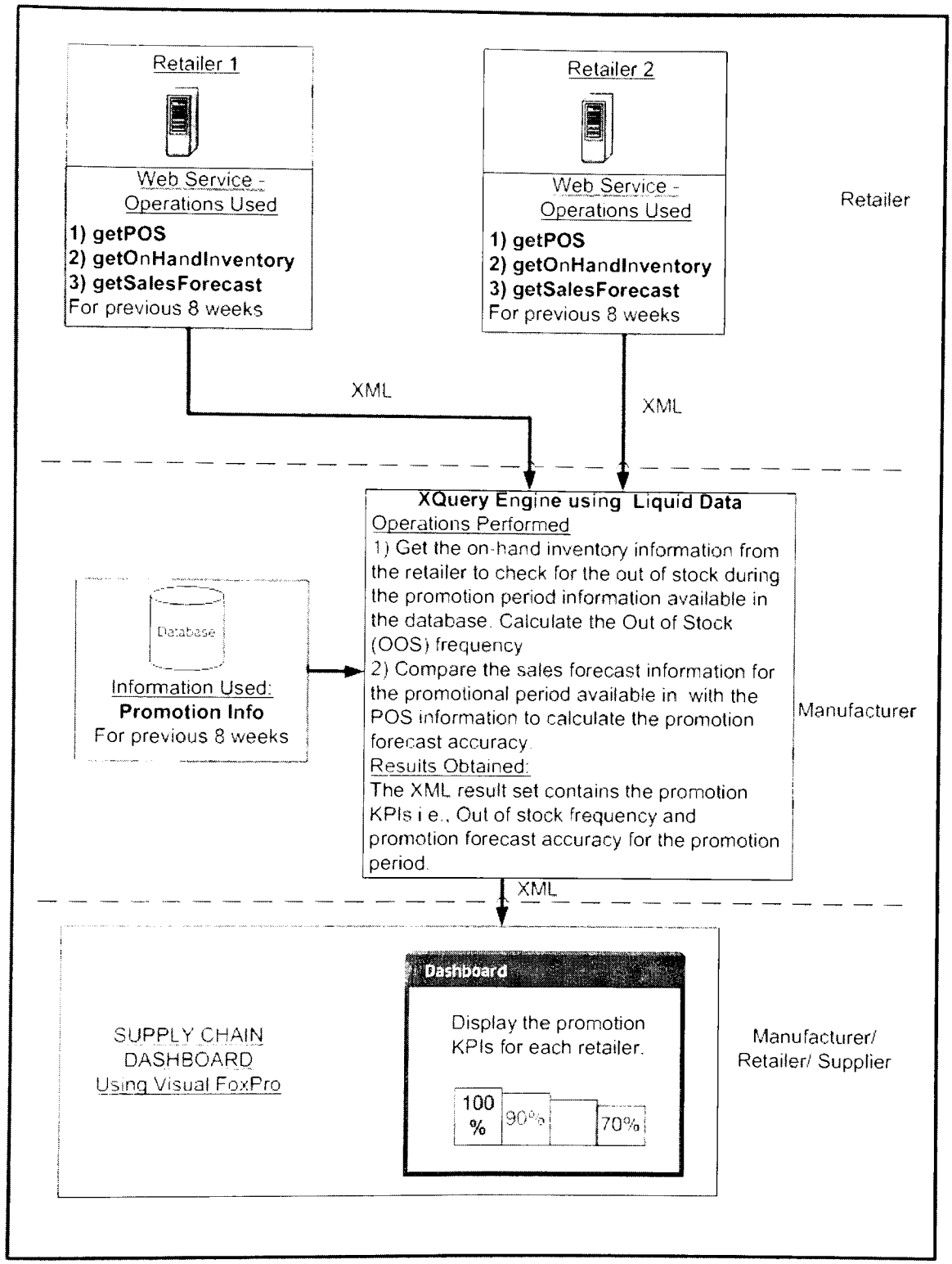

Figure 4.3. Scenario 2a – Information flow for exception on promotions.

4.4 Scenario 2b – Identifying Promotion Plan Exceptions

In this scenario, the manufacturer is interested in checking the conflicts in the promotions forecast for the next eight weekly periods. If the promotion forecast for both the retailers for an item is planned for the same time bucket, the manufacturer runs the risk of not meeting the customer demand as a result of a possible surge in demand. These conflicts are identified as **exceptions**. The manufacturer can then work on resolving the exceptions by requesting the retailers to have the promotion events at different time periods, thereby allowing the manufacturer enough time to meet each order. This would mean that the surge in the demand forecasted for a specific time period is better managed [7].

4.4.1 Data Sources Used

The data sources for this scenario are as follows:
1) An XML file, "**promotionsforecast**," which holds the information about the sales forecast for the next eight time buckets, is considered as an equivalent to a **getPromotions forecast webservice** response from each retailer.

4.4.2 Processing Done

An XQuery program is written to compare the promotion information for each retailer. The promotion information is available in XML files and the XQuery program is coded to generate exceptions using a "Has Exception" flag. The start date of each XML file is compared to check if the nodes have promotions. If there are promotions planned for the same start date, the XQuery program returns a value of "true" to the "Has Exception" field of the output XML file.

4.4.3 Results Obtained

The output from the XQuery is XML data which can be made available as a Web Service. This Web Service outputs an XML response that pertains to an XML Schema for the result. The XML Schema contains the following elements:

1) Seller ID
2) Buyer ID
3) Item ID
4) Has Exception
5) Promotion Type ID
6) Promotion Description

Figure 4.4 depicts the information flow in this scenario.

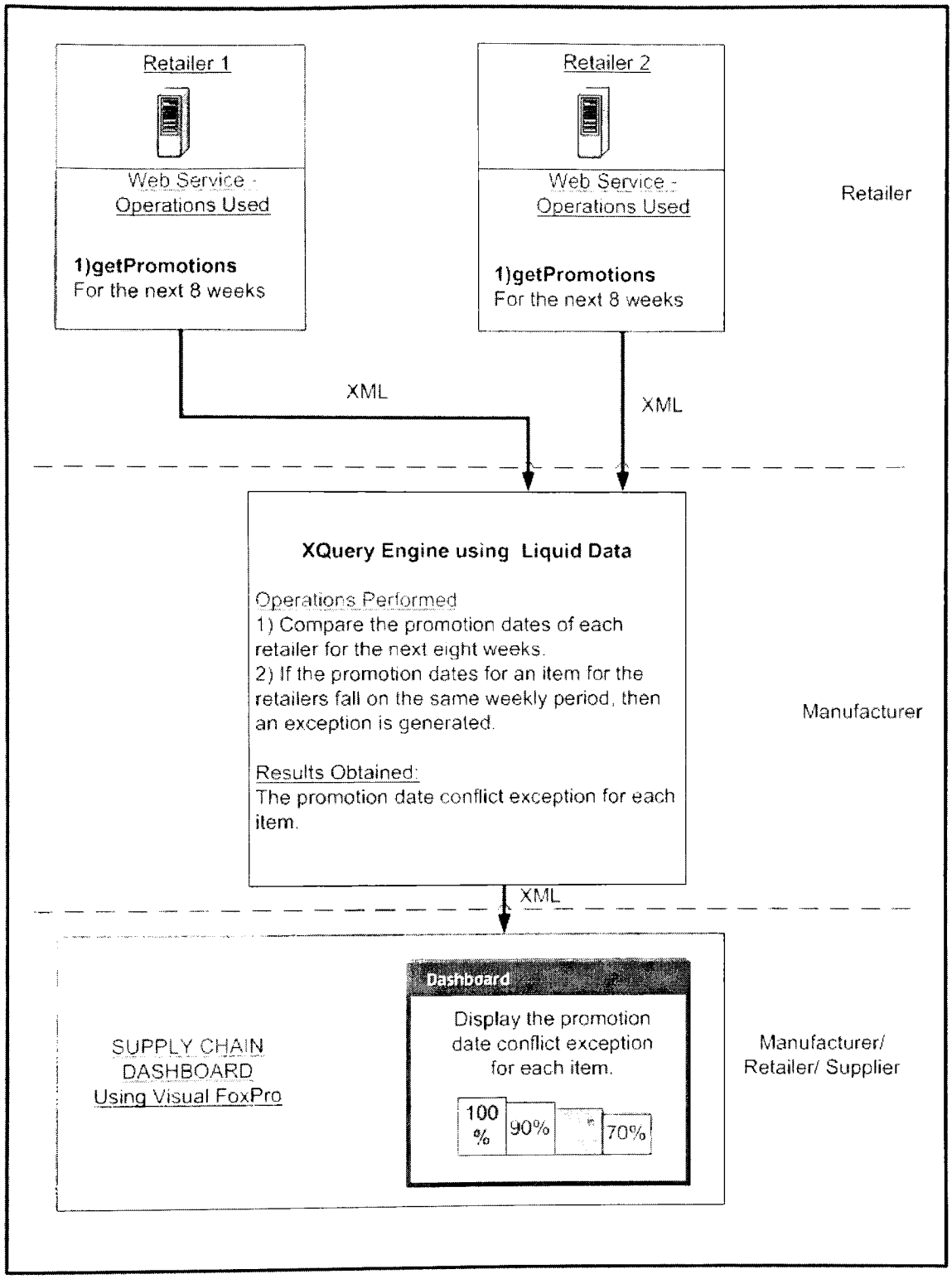

Figure 4.4. Scenario 2b – Information flow for generating exceptions on promotion events.

4.5 Scenario 3a – Calculating Order Forecast Accuracy KPI

In this scenario, the manufacturer is interested in verifying the order forecast accuracy for the last eight weekly periods. The order forecast data for the retailers for the previous eight weeks are consolidated and contrasted with the actual orders available with the manufacturer to calculate the order forecast accuracy. The order forecast accuracy is a key performance indicator (KPI) that can be used to refine future forecasts.

4.5.1 Data Sources Used

The data sources for this scenario are as follows:
1) A **CSV file**, "actual orders," contains the information about the orders for each item for the previous eight weekly periods.
2) A **SQL server database** of the Manufacturer, "**CPFR-Manufacturer**," contains the Sales forecast information; the "orderforecast" table contains the order forecast data required for this scenario.

4.5.2 Processing Done

An XQuery program is written to compare the order information available in the CSV file for each retailer with the order forecast data for the previous eight weeks in the manufacturer's database for that retailer to calculate the order forecast accuracy for each time period. This Key performance indicator is calculated using the formula mentioned below.
order forecast accuracy in %
= 1 - [(actual order quantity - order forecast quantity)/actual order quantity]*100

4.5.3 Results Obtained

The output from the XQuery is XML data which can be made available as a Web Service. This Web Service outputs an XML response that pertains to an XML Schema for the result. The XML Schema contains the following elements:

10) Seller ID

11) Buyer ID

12) Start date

13) End date

14) Actual Order Quantity

15) Order Forecast Quantity

16) Order Forecast Accuracy

Figure 4.5 depicts the information flow in this scenario.

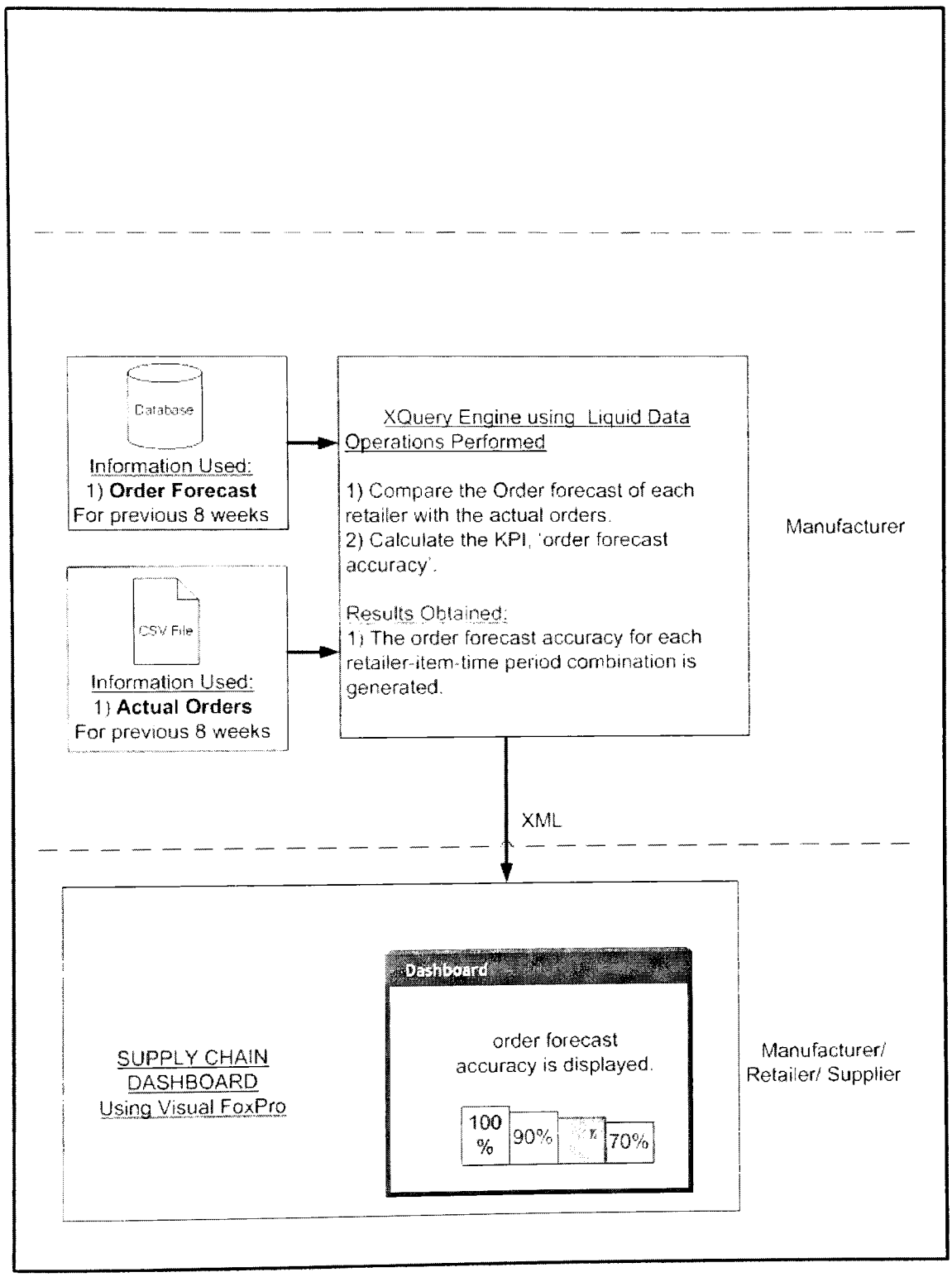

Figure 4.5. Scenario 3a – Information flow for calculating the Order forecast accuracy.

4.6 Scenario 3b – Identifying Order Forecast/Delivery Constraint Exceptions

In this scenario, the manufacturer is interested in checking the variation in the order forecast for the next eight weekly periods and ensuring that the order forecasts for each week can be met. The variations (more than 10%) in the future sales forecasts for the two retailers and the manufacturer are identified as **forecast variation exceptions** for further negotiation in arriving at a final sales forecast. The aggregated order forecast is compared to the "**Available To Promise**" (ATP) system to generate exceptions when there is not enough inventory for promising orders during the particular weekly period. The ATP systems are used to see how much inventory or projected inventory is not committed to customer orders and is therefore available for order promising. ATP is normally calculated from the Master Production Schedule and is maintained as a tool for customer order promising. With ATP the uncommitted portion of a company's inventory and the expected receipts, maintained in the master schedule or in purchase order management, are used to support customer order promising [1].

4.6.1 Data Sources Used

The data sources for this scenario are as follows:

1) An XML file, "**orderforecast**," which holds the information about the order forecast for the next eight time buckets, is considered as an equivalent to a **getOrderforecast webservice** response from each retailer.

2) A SQL **server database** of the Manufacturer, "CPFR-Manufacturer," contains the Order forecast information; the "orderforecast" table contains the order forecast data required for this scenario.

4.6.2 Processing Done

An XQuery program is written to aggregate the order forecast quantity available in the XML file for the retailers and compare it with the ATP quantity available in the CSV file for each weekly period. If the order forecasts exceed the ATP quantity for a weekly time period, the

system generates a delivery constraint exception. If the aggregate quantity is less than the STP quantity, then order information available in the XML file for each retailer is compared with the data in the manufacturer's database for each retailer. If the order forecast variation is more than 10%, the system flags the output for that time bucket to contain an exception. An exception can be defined as a variation in the order forecasts that is too large, such that the business partners must collaborate and resolve to arrive at a common order forecast.

4.6.3 Results Obtained

The output from the XQuery is XML data which can be made available as a Web Service. This Web Service outputs an XML response that pertains to an XML Schema for the result. The XML Schema contains the following elements:

7) Seller ID
8) Buyer ID
9) Seller forecast quantity
10) Buyer forecast quantity
11) Has Exception
12) Exception Type ID
13) Difference

Figure 4.6 depicts the information flow in this scenario.

These CPFR scenarios are implemented using a commercially available XQuery engine called "Liquid Data." The steps involved in adding data sources, creating XQuery expressions, and using the resulting Web Service in Visual FoxPro are explained in the next chapter.

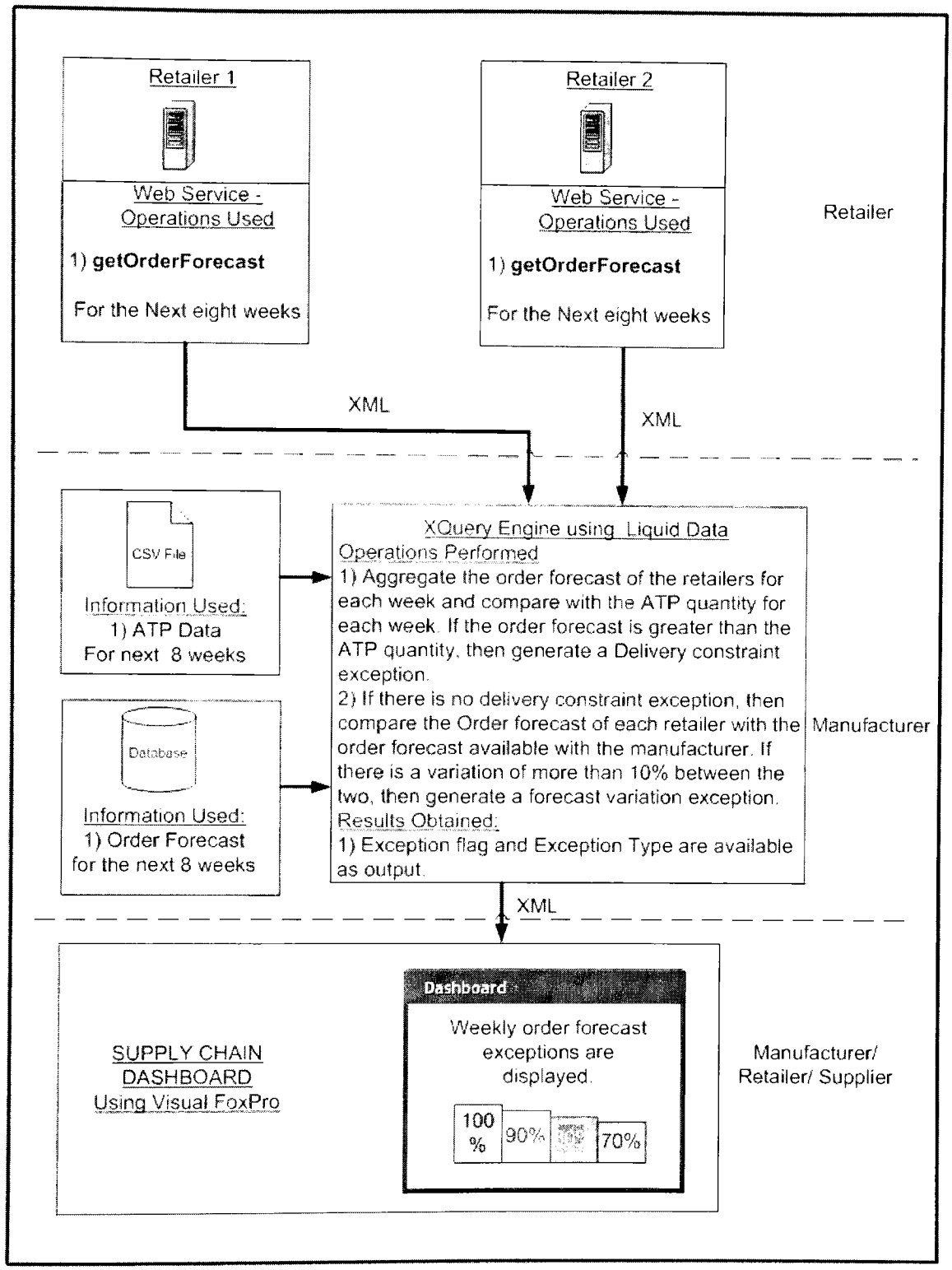

Figure 4.6. Scenario 3b – Information flow for identifying exceptions in order forecast.

5. Developing a CPFR Dashboard

Collaborative planning, forecasting and replenishment (CPFR) aims at collaboration among the participating business partners in a supply chain to arrive at one common order forecast. To achieve this, the information such as sales forecast, promotion event information, point of sale data, receipts data, on-hand inventory data and order forecast data are made available by the business partners at the customer end to all the business partners upstream in a supply chain. This enables a demand-driven business model where all the participating members in a supply chain function to meet customer demand [7].

A CPFR application at the manufacturer's end is intended to fetch data from the business partners and aggregate it with the data at the manufacturer end to check for exceptions in the sales and order forecasts and to notify the managers to take actions. In the current economy, information sharing is done using Web Services. Hence there is a need to aggregate data available in a Web Service with the data available in relational databases and Excel spreadsheets [7].

Figure 5.1 depicts a data aggregation example where information from two retailers is aggregated with information from databases and CSV files at the manufacturer's end. The XQuery application communicates with the Web Services at the retailers' end (retailer1 and retailer 2) and obtains the forecast, promotion and inventory information for performing KPI calculations.

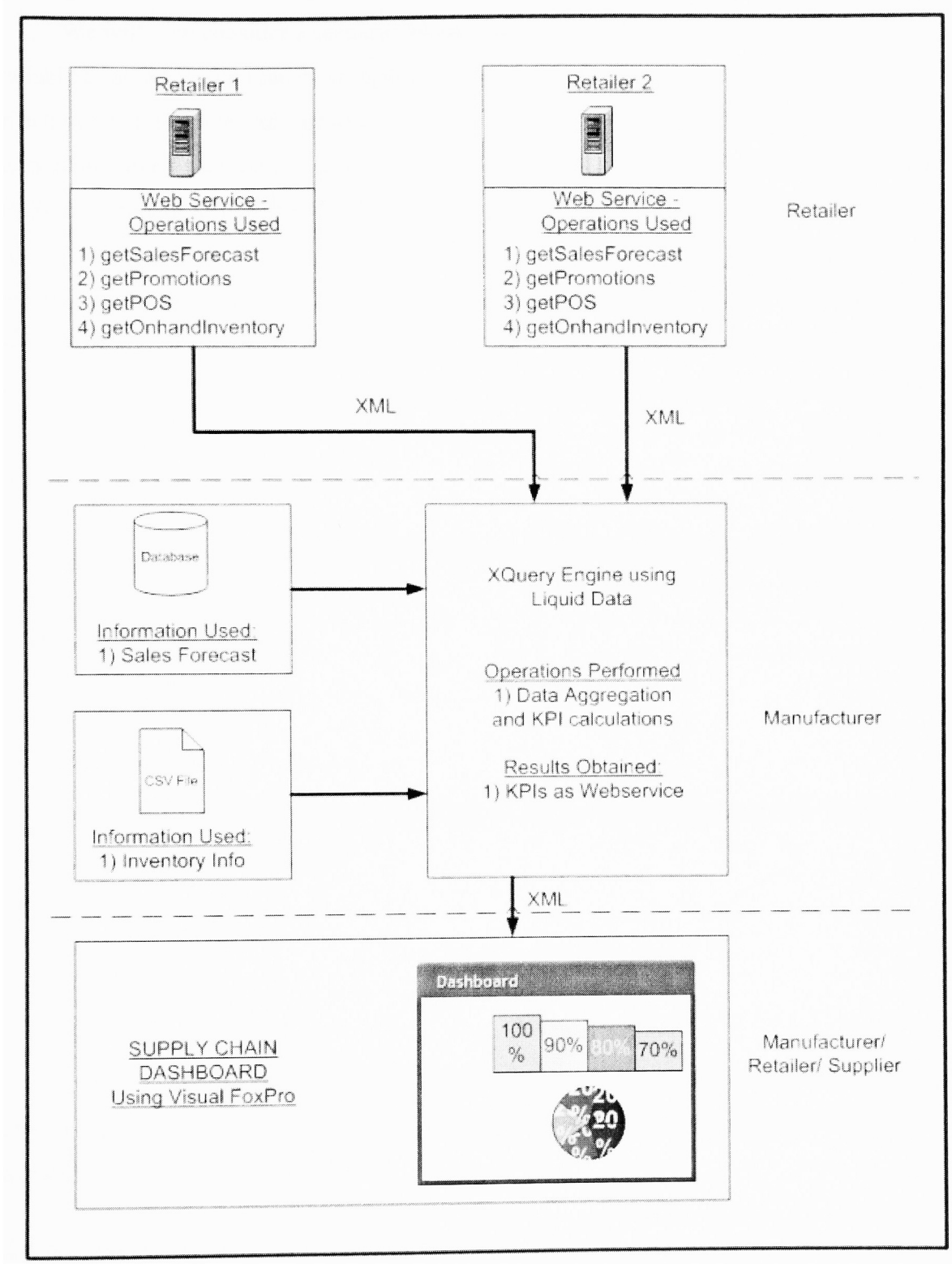

Figure 5.1. Architecture of the CPFR dashboard application.

We will now consider a scenario where order forecast information from the retailers is available as an XML file (can be compared to a Web Service response). An XQuery program is written to fetch this data and aggregate it with the actual order data available in a CSV file from the manufacturer. The weekly order forecast accuracy, a Key performance indicator (KPI) in a CPFR environment, is calculated using XQuery and made available as a Web Service. This Web Service is then consumed by a visual FoxPro application to generate graphs for use in an executive dashboard. The XQuery programs are developed using an XQuery software engine called "Liquid data." Please refer to Appendix II for installation and setup instructions for Liquid data. We will now present the steps involved in developing this application in a tutorial format.

5.1 Identifying the Data Sources

The weekly order forecast information is available in an XML file. The information that is available in this XML is

1) Item ID
2) Week Number
3) Forecast Quantity

The actual order information is available in a Excel spreadsheet as a CSV file. The information available in this file is

1) Item ID
2) Week number
3) Order Quantity

An XML Schema is developed for the information available in the CSV file. Hence the Schema also contains a "repeatable" node called "order" containing the elements "itemid," "weeknum" and "orders."

5.2 Configuring the Data Sources

We will use BEA Liquid data as an XQuery engine. Liquid data requires that the data sources be configured using its Admin module before calling the data source in the XQuery code. Start

the Liquid data server using Start->Programs->BEA WebLogic Platform->BEA Liquid data for WebLogic 8.1SP3->Liquid Data Samples->Samples Server.

We will now configure the data sources. Open Liquid data by clicking on Start->Programs->BEA WebLogic Platform->BEA Liquid data for WebLogic 8.1SP3->Liquid Data Samples-> Admin Console. The admin console opens up in a browser window. Enter the username and password as system and security. After logging in, locate and click on the Liquid data link in the menu tree in the left navigation window.

Now, click on Configuration->Data Sources-> XML Files. This lists the existing XML files that are configured as valid XML data sources. We will now create a new XML file source. Click on "Configure a new XML file data source." It opens up a form as shown in Figure 5.2.

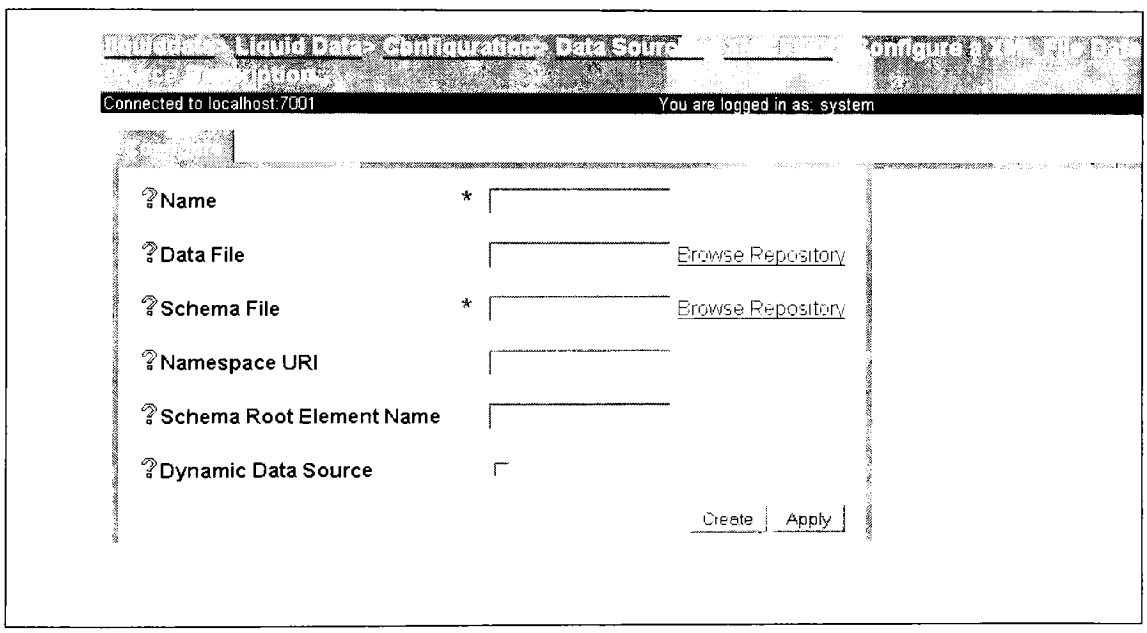

Figure 5.2. XML file data source configuration.

Enter the following information:

Name : orderforecast

Data File : orderforecast.xml

Schema File : orderforecast.xsd

We need to make sure the files "orderforecast.xml" and "orderforecast.xsd" are available in C:/bea/weblogic81/samples/domains/liquiddata/ldrepository/xml_files and C:/bea/weblogic81/samples/domains/liquiddata/ldrepository/schemas respectively.
Click on Create. This creates an XML file data source and displays "orderforecast" as a data source in the list.

A CSV file is a delimited file. To create a CSV data source, click on Configuration->Data Sources-> Delimited Files. Click on "Configure a new Delimited file data source description." Fill in the form using the information available as in Figure 5.3 below.

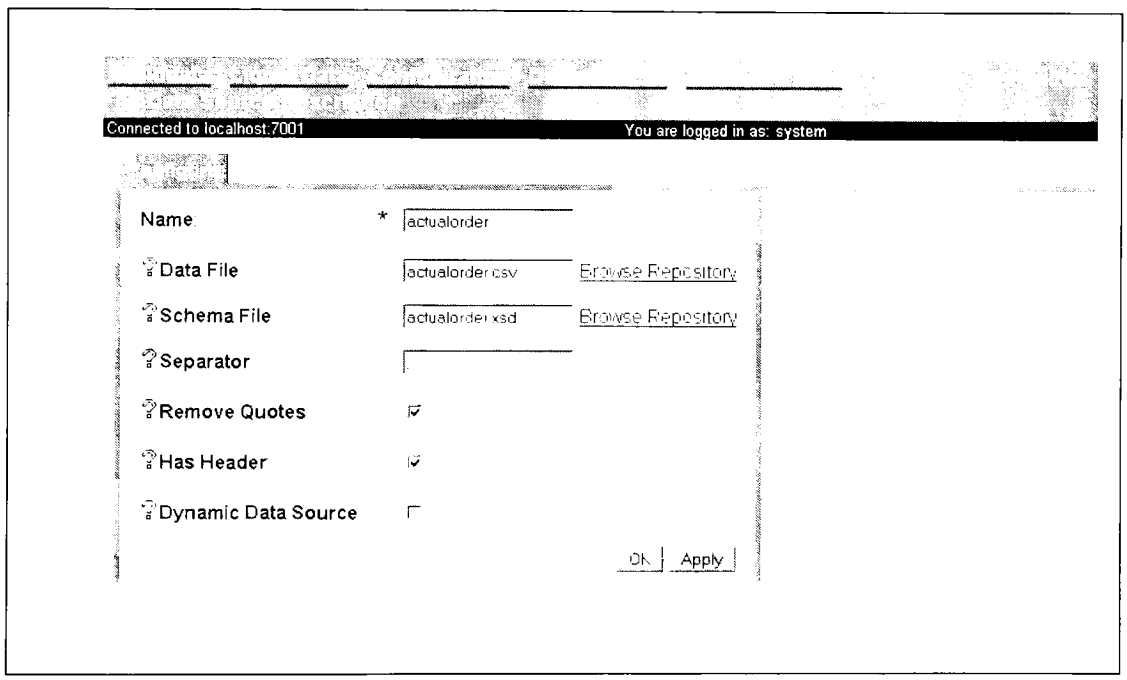

Figure 5.3. Configuring a delimited file data source description.

We need to make sure the files "actualorder.csv" and "actualorder.xsd" are available in C:/bea/weblogic81/samples/domains/liquiddata/ldrepository/delimited_files and C:/bea/weblogic81/samples/domains/liquiddata/ldrepository/schemas respectively.

5.3 Developing a Target XML Schema

The target XML Schema is developed that contains the structure of the target document. Since the target XML would contain the order forecast accuracy information for different time buckets, we will create a repeatable node called "kpi." The "kpi" node contains the following elements:

1) Item ID
2) Week Number
3) Forecast Quantity
4) Actual Order Quantity
5) Order Forecast Accuracy

Save the target Schema as "thesis1.xsd."

5.4 Developing XQuery Code for Data Aggregation

We will now develop an XQuery program to aggregate the data available in the XML file and the CSV file. Liquid data's Data View Builder has the functionality to write XQuery code and to test it. We will now use Data View Builder to write our XQuery code. To open Data View Builder, click on Start->Programs->BEA WebLogic Platform->BEA Liquid data for WebLogic 8.1SP3->Data View Builder. A login window appears. Click on OK button without entering the username and password.

Data View Builder screen has a left navigation window containing the list of data sources as shown in Figure 5.4.The left navigation window also has a menu containing the list of all the XQuery functions and operators that can be used in the program, as shown in Figure 5.5. The center screen is a graphical interface where the data sources can be pulled into it and mapped to elements in other data sources [28].

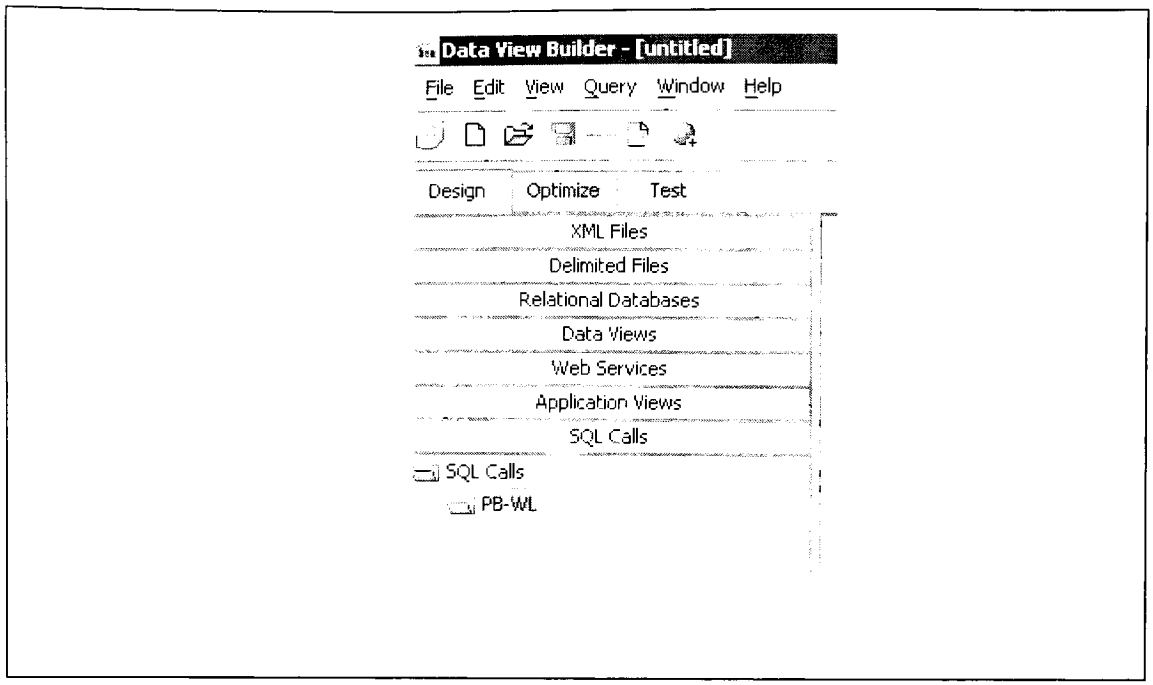

Figure 5.4. Left navigation screen displaying data sources.

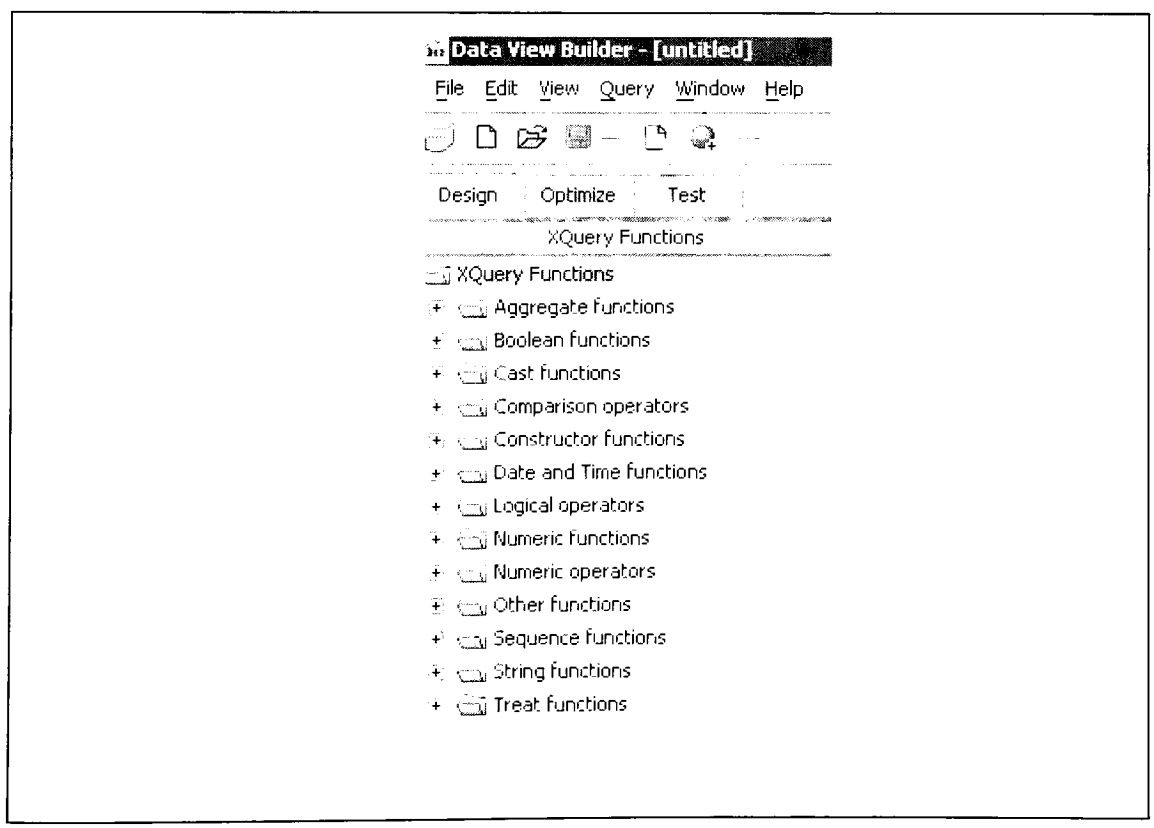

Figure 5.5. Left navigation screen displaying functions.

Before writing the XQuery code, Data View Builder requires that we have the data sources and the target Schema in the work environment. To bring the data sources to the work environment, pull the required data sources (the XML data source, orderforecast and the delimited file data source, actualorder) to the center window pane. To open the target Schema, click on the top menu, File-> Set Target Schema. Locate the target Schema and select it. The target Schema opens up the right window pane. The elements of the Schema are available in a tree view. You can expand the tree to view all the elements in the Schema [28].

This XQuery performs the function of aggregating the data from the two different data sources. Table 5.1 shows the parameters in different formats from the two data sources mapped to the target XML Schema.

Table 5.1 *Mapping the Data fields to the Target Schema*

Data Source	**Data Source Type**	**Source Field**	**Target Field**
orderforecast	XML File	Itemid	Itemid
orderforecast	XML File	Weeknum	weeknum
orderforecast	XML File	quantity	forecast
actualorder	CSV File	quantity	actual
orderforecast and actualorder	-NA-	Aggregated the data from the two data sources and calculated the order forecast accuracy. 1 - [(actualorders-orderforecast)/actual orders]*100	accuracy

An XQuery code as shown below can be used for aggregating the data and also to calculate the order forecast accuracy, a KPI in a CPFR environment.
The order forecast accuracy can be calculated as follows:

Order forecast accuracy in % = 1 - [(actual order quantity - order forecast quantity)/actual order quantity]*100

We will use the XQuery operators such as -, div, * and cast_as_xs:float and cast_as_cs:integer to calculate the key performance indicator.

```
<results>
    {
    for $orderforecast.forecast_1 in document("orderforecast")/orderforecast/forecast
    for $actualorder.Orders_2 in document("actualorder")/OrderInfo/Orders
    let $v_3 := $actualorder.Orders_2/quantity - $orderforecast.forecast_1/quantity
    let $cast_as_xs:float_4 := cast as xs:float($v_3)
    let $cast_as_xs:float2_5 := cast as xs:float($actualorder.Orders_2/quantity)
    let $div_6 := $cast_as_xs:float_4 div $cast_as_xs:float2_5
    let $v2_7 := 1 - $div_6
    let $v_8 := $v2_7 * 100
    where ($orderforecast.forecast_1/weeknum  eq  $actualorder.Orders_2/weeknum)
    return
    <kpi>
        <itemid>{ xf:data($orderforecast.forecast_1/itemid) }</itemid>
        <weeknum>{ xf:data($orderforecast.forecast_1/weeknum) }</weeknum>
        <forecast>{ xf:data($orderforecast.forecast_1/quantity) }</forecast>
        <actual>{ xf:data($actualorder.Orders_2/quantity) }</actual>
        <accuracy>{ cast as xs:decimal(cast as xs:integer($v_8)) }</accuracy>
    </kpi>
    }
</results>
```

Save this XQuery as "thesis1.xq." To test this XQuery, move to the Test Pane and click on Query->Run Query in the top menu. Since our XQuery program does not need an input parameter, the Data View Builder does not wait for the input from the user and proceeds to execute the XQuery. The results of the XQuery are displayed in the right pane. Figure 5.6 shows the resulting XML that is generated by our XQuery. The results show five child nodes displaying

the information for five time buckets, i.e., week 1, 2, 3, 4 and 5. The field "accuracy" is the new information that is calculated by aggregating information from the two data sources and performing arithmetic operations on them. "Accuracy" refers to the order forecast accuracy information, a KPI for each time bucket.

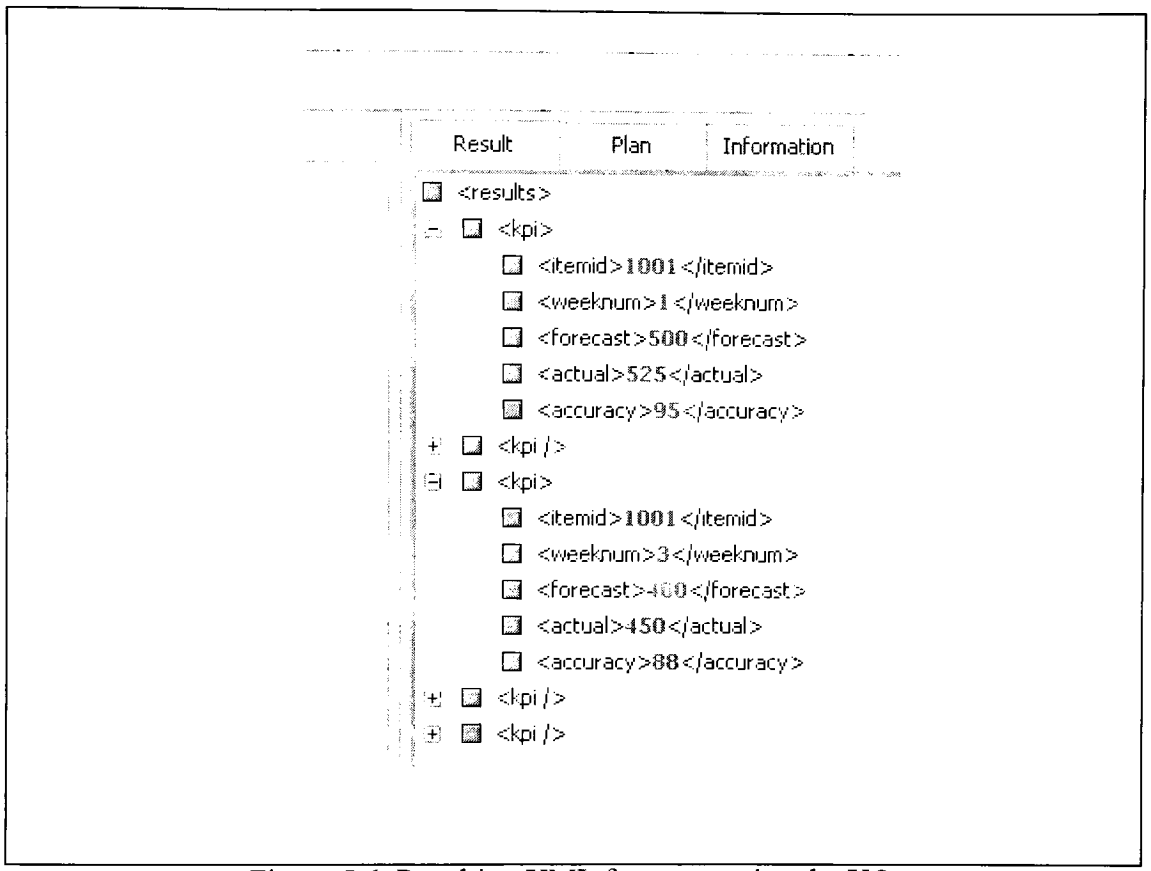

Figure 5.6. Resulting XML from executing the XQuery

This key performance indicator information can be used by all participating business partners, as this KPI would help the managers at each end in a supply chain to measure the performance of the entire supply chain. If this information is available as a Web Service, it can be easily called by any web application for use by each business partner.

The XQuery program can be made available as a Web Service using the Liquid data. To achieve this, Open the Liquid data's Admin Console by clicking on Start->Programs->BEA WebLogic Platform->BEA Liquid data for WebLogic 8.1SP3->Liquid Data Samples-> Admin Console. After logging in and getting into the Liquid data page, locate the Stored Queries list by

clicking on Configuration->Stored Queries in the top menu. Our XQuery program "thesis1" is listed. Click on the "Generate Web Service" link to generate a Web Service for this XQuery. The WSDL document location is displayed when the Web Service is successfully generated.

5.5 Developing the Dashboard Using Visual FoxPro

The XQuery that we have developed is now available as a Web Service; hence we can use an application such as Visual FoxPro to call the Web Service, and we can use the XML data to develop dashboard applications with richer graphics for better presentation of information.

A FoxPro form is developed as shown in Figure 5. The form has controls to call a Web Service, to convert the XML data into a cursor, and to use the data from the cursor to display graphs. In this application we will use the XML data and create a graph displaying the order forecast accuracy for each week. Before invoking a Web Service in Visual FoxPro, it is required that the Web Service is registered. A Web Service is registered using its WSDL document location. The "thesis1" Web Service location is http://imert123:7001/liquiddata/thesis1/webservice?WSDL.

To register the Web Service in Visual FoxPro, click on Tools->Toolbox. A small toolbox window pops up displaying the menu items available in the toolbox. Locate the "My Webservices" link and click on it. This displays the list of registered Web Services. To register our "thesis1" Web Service, click on the "Register' link. When asked for a WSDL URL location, type http://imert123:7001/liquiddata/thesis1/webservice?WSDL and click on the Register button. The Web Service appears in the list of available registered Web Services. To use this Web Service in the code, drag and drop the Web Service into the editor. Visual FoxPro generates code for invoking the registered Web Service [27].

To invoke the "thesis1" Web Service, use the following code:

```
LOCAL lothesis1ServicePort AS "XML Web Service"
```

```
LOCAL loException, lcErrorMsg, loWSHandler

loWSHandler =
NEWOBJECT("WSHandler,"IIF(VERSION(2)=0,","HOME()+"FFC\")+"_ws3client.vcx")
lothesis1ServicePort =
loWSHandler.SetupClient("http://imert123:7001/liquiddata/thesis1/webservice?WSDL,"
"thesis1Service," "thesis1ServicePort")
        leResult = lothesis1ServicePort.thesis1(")
        XMLTOCURSOR(leResult.ITEM(0).parentnode.childnodes(0).XML,"cur")
```

The XMLTOCURSOR() function converts the resulting XML from a Web Service response into a Visual FoxPro cursor.

Visual FoxPro provides the functionality to access and use the Microsoft ActiveX controls for displaying rich graphics controls such as graphs and progress bars. We will use a graph to display the order forecast accuracy values for each time bucket. The following code will display a graph using the data stored in the cursor we just created:

```
SELECT * from cur
WITH THISFORM._Autograph
        DIMENSION .aDataFields[1]
        .aDataFields[1] = "accuracy"
        .cCategoryField = "weeknum"
        .cTitle = "Weekly Order Forecast Accuracy Graph"
        .lAddLegend = .T.
        .lAddTitle = .T.
        .nChartType = 11                    && Chart Type
        .nChartSubType = 1                  && Chart SubType
        .lSeriesByRow = .F.                 && Series by Row (.T.), by Column (.F.)
        .nAction = 0
        .cGraphPrevClass = "graphpreview"  && Class containing preview form
```

```
        .cDefNewField = "olegraph"          &&default field name in new table
        .MakeOutput()
ENDWITH
```

Figure 5.7 shows the Dashboard application that generates a graph using Visual FoxPro.

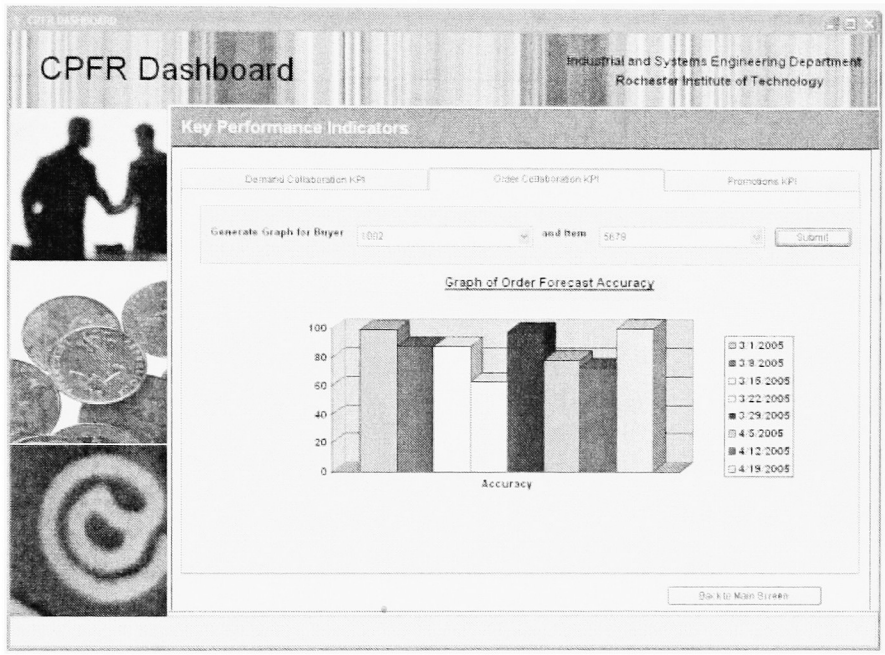

Figure 5.7. CPFR Executive Dashboard.

We have now successfully aggregated the data from two different data sources using XQuery and used the resulting XML to generate a dashboard using Visual FoxPro.

6. Conclusion

The primary focus of this research work was to evaluate XQuery as a tool for supply chain collaboration. Supply chain collaboration has gained significance over the last few years as a result of increased focus on reducing costs to remain competitive in an increasingly customer-driven market. Reducing cost in a supply chain would mean reducing the inventory levels and improving the forecasts at each business partner end. This would also result in improved service levels and hence would facilitate an efficient supply chain. Over the years, newer supply chain concepts and the emergence of the Internet paved the way for new opportunities for companies to collaborate effectively. As companies start to collaborate, they increasingly face challenges in accessing and aggregating data from disparate data sources in real time. Aggregating real-time data is critical in a supply chain, as a consolidated view of the real-time data will assist in effectively monitoring the supply chain. XQuery has emerged as the tool to access and aggregate data from data sources as varied as data from a Web Service, a relational database and an Excel spreadsheet. XQuery as a programming language provides operators and functions to aggregate data from different data sources and to provide a consolidated view of the data that can be used to monitor and measure the performance of a supply chain [9].

In this study, we reviewed the basic features of a supply chain. A supply chain is the series of links and shared processes that exist between suppliers and customers. These links and processes involve all activities from the acquisition of raw materials to the delivery of finished goods to the end consumer. We distinguished a simple supply chain from an extended supply chain. An extended supply chain includes partners such as third party logistics providers who contribute to the supply chain indirectly. Supply chains are divided into two types based on the demand pattern for products that come out of a supply chain. The first type, an efficient supply chain, is suited for functional products for which the demand has minimal variations and is thus more predictable. Hence the primary purpose to choose an efficient supply chain is to supply products at the lowest possible cost. This would mean maintaining the lowest possible inventory and reducing lead time as long as it does not increase the cost. The other type, a responsive supply chain, is suited for innovative products which have large demand variations (as they won't have sufficient historical data to forecast demand), and for which the demand is therefore

less predictable. A responsive supply chain would call for deployment of significant buffer stocks and lowest lead time as to avoid out-of-stock situations [1]. The reasons for demand variations were explored, specifically targeting the "Bullwhip" effect as a significant factor for demand variation across the supply chain. We discussed Supply chain management (SCM) as a concept to manage supply chains effectively and efficiently, thereby increasing customer value and retaining a competitive advantage. The SCM systems were compared with ERP systems with respect to complexity and speed of processing requests. Typically ERP systems hold enterprise-wide data, of which a part becomes input for SCM systems [14].

In this study we aimed at developing a software application architecture for supply chain collaboration by aggregating data using XQuery. For this purpose, we considered a supply chain collaboration concept called "Collaborative Planning, Forecasting and Replenishment (CPFR). CPFR is a set of standards defined by Voluntary Interindustry Commerce Standards (VICS) that formalizes the processes between two trading partners used to agree upon a joint plan and forecast, monitor success through replenishment, and recognize and respond to any exceptions. VICS proposes a 9-step process model, the result of which is a common order forecast generated by resolving the exceptions/mismatch in sales and order forecast among business partners. We discussed the conceptual differences between a CPFR system and its predecessor, a Vendor managed inventory (VMI) system. We discussed key performance indicators (KPI), which are quantifiable measurements for measuring the performance of a CPFR system. The KPIs specific to CPFR were identified for use in this study. Executive dashboards, which are highly customized software applications, were discussed as a tool to graphically represent the KPIs. We discussed the significance of dashboards as effective tools for enterprises to have a "Big Picture" view of the performance of the supply chain. We recognized that the performance of a CPFR process which is implemented can be measured using KPIs and presented graphically for quick and intelligent decision making [7].

In this study, we identified the technologies that can be used to implement the CPFR system. We realized that data in a supply chain system exists in various forms and through a variety of data sources. We recognized XML as a standard and common data structure for sharing data between disparate systems. The elements and attributes form the components of the

XML document. The structure of the XML document is defined using standard specifications recommended by the World Wide Web (W3C) consortium. We identified XML Schema, the latest specification, as an effective way to represent the structure of the XML document owing to its extensive support for a variety of data types as compared to its predecessor, DTD. As enterprises recognized XML as a standard for data, and the Internet as a standard means of data communication, new technologies emerged as means for vendor-independent and technology-neutral software applications to communicate with each other. Web Services are software applications that can interact with other applications over the Internet. We discussed the architecture of a Web Service and identified the key components of a Web Service. XML, Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI) are components that are defined as separate standards for universal data definition and communication. We recognized that Web Services can be used as a means for enterprises to communicate for supply chain collaboration. As CPFR systems use data that exist in disparate data sources, we explored XQuery, a W3C specification to aggregate data and perform complex calculations on it. XQuery is a language for data aggregation. The set of operators and functions gives a powerful combination of support for a variety of data types and data sources, and functionality to perform complex arithmetic calculations. We differentiated XQuery from XPATH, its predecessor, and identified significant advantages in XQuery for data manipulation and aggregation [6].

Having identified XQuery as the language for data aggregation, several CPFR scenarios were developed to prove the data aggregation for supply chain collaboration. Three of the scenarios (1a, 2a and 3a) consolidate data into key performance indicators while the other scenarios (1b, 2b and 3b) are generating exceptions. These scenarios involve the data exchange between a manufacturer and two retailers. The retailers share their information on sales forecasts for the next eight weeks, order forecasts for the next eight weeks, POS data for the previous eight weeks, and promotion event information for the next eight weeks. The primary data sources are Web Services and their equivalent XML files, SQL server data bases and CSV/Excel files. The retailers share information using Web Services, which the manufacturer uses to compare the sales forecasts and order forecasts existing in the relational database.

We discussed four exceptions to be identified. Two of the exceptions, the sales and order forecast exceptions, are calculated by comparing the sales and order forecasts of the retailer with that of the manufacturer. The comparison will result in exceptions in forecasts when the variation in forecasts between the retailer's and manufacturer's data is over 10%. The third exception is a delivery constraint exception. To identify this, the order forecast from the retailer is compared with the available to promise (ATP) system to verify if the future orders can be promised. The available to promise (ATP) system holds information on the quantity of a certain product the manufacturer can promise for delivery at a certain time. This information exists in a CSV/Excel file. The exception flag is set when the ATP quantity is less than the order forecast for a certain period. This flag indicates that a future order may not be fulfilled due to supply constraints. The fourth exception is the promotion conflict exception. To identify this, we compare the promotion information of the two retailers to see if we encounter the promotion event at the same period of time. This information would prove very critical as the manufacturer would have to plan well ahead of time to meet this surge in demand.

We also identified three KPIs to be measured: the sales forecast accuracy, out of stock (OOS) frequency and order forecast accuracy. Sales and order forecast accuracy measure the performance of the forecasts, whereas, the out of stock frequency measures the performance of a promotion event.

The information from the different data sources is made available as XML data. The XML engine used to convert the data sources to XML and to process the XQuery code is "Liquid Data" by BEA systems. We identified the data sources used, the processing done and the results obtained for each of the scenarios. The resultant XML file contains the KPIs and exception flags which are of importance to us.

We recognized that Visual FoxPro provides the functionality to consume a Web Service and also provides a variety of graphical tools to develop a dashboard. We developed a software application using Visual FoxPro to display the exceptions and KPIs graphically. Dashboards proved very effective as it was easier to identify the trend in forecasts and also locate the

exceptions for the eight-week period. Various graphs were developed to display the exceptions for each retailer.

Throughout this study, emphasis was put upon having the application architecture as modular as possible. The center point of this research work is exploring XQuery and evaluating it as a tool for supply chain collaboration. We identified supply chain collaboration scenarios from CPFR, a recognized and proven supply chain collaboration concept that defines standards for systematically identifying and resolving forecast mismatches among business partners so as to arrive at common forecasts. These scenarios require the aggregation of data from varied data sources—i.e., XML data from business partners, relational data from databases, and CSV data. We successfully developed XQuery code to aggregate this data and perform several arithmetic calculations to arrive at the CPFR KPIs. We were able to develop a dashboard application using Visual FoxPro for presenting the exception information and KPIs graphically. This enables enterprises to monitor the supply chain in real time and to take quick decisions to resolve any conflicts. Enterprises in a supply chain will find XQuery to be a powerful tool for aggregating their data, as they would have flexibility in integrating data from a variety of data sources and can integrate with the other business partners independent of the technology their applications run on.

References

[1] John T. Mentzer, William DeWitt, James S. Keebler, Soonhong Min, Nancy W. Nix, Carlo D. Smith, and Zach G. Zacharia, 2001, "Defining supply chain management," *Journal of Business Logistics*, Vol. 22, No. 2, p. 18.

[2] Hau L. Lee, V. Padmanabhan, and Seungjin Whang, 1997, "The bullwhip effect in supply chains," *Sloan Management Review*, Vol. 38 (Spring), 93-102.

[3] Covalent Works, "Overview of EDI benefits and drawbacks," 2004, http://www.covalentworks.com/what-is-edi.asp

[4] IBM Corporation, "Get a grounding on the basic concepts," 2003, http://www-106.ibm.com/developerworks/library/ws-starthere.html

[5] John Taylor, "Thoughts from the Integration Consortium: Enterprise Information Integration: A new definition," Integration Consortium, 2004, http://www.dmreview.com/article_sub.cfm?articleId=1009669

[6] Michael Brundage, *XQuery: The XML Query language*, Addison Wesley, 2004, Chapter 1.

[7] Voluntary Industry Commerce Standards, "Collaborative Planning, Forecasting and Replenishment: Version 2.0," 2002, http://www.vics.org/committees/cpfr/voluntary_v2/CPFR_Tabs_061802.pdf

[8] Data Direct Technologies, "XQuery tutorial: Building XQuery based aggregation and reporting applications," 2005, http://www.stylusstudio.com/xquery_tutorial.html

[9] David Anderson and Hau Lee, "The Internet enabled supply chain: From the first click to the last mile," 2000, http://www.ascet.com/documents.asp?d_id=199

[10] Ravi Trivedi, Web services: Understanding XML and XML Schema, Jupitermedia Corporation, 2003, http://www.developer.com/services/article.php/2195981

[11] Bluebill Advisors, Inc., "What is Enterprise Information Integration?" The Gilbane Report, 2004, http://www.gilbane.com/artpdf/GR12.6.pdf

[12] John Babb, "Supply Chain Management: An executive reference paper," Clarkston Consulting, 2000, http://www.clarkstonconsulting.com/WhitePaper/SupplyChainMgmtPaper.pdf

[13] Michael Hugos, "Essentials of Supply Chain Management," pp. 5-11, December 2002.

[14] Arnold, J.R. Tony.,2003, *Introduction to Materials Management*, 3rd edition, p. 5.

[15] Hau L. Lee, V. Padmanabhan, and Seungjin Whang, 1997, "The bullwhip effect in supply chains," *Sloan Management Review 38* (Spring), 93-102.

[16] Scott Stephens, 2001, *Supply Chain Operations Reference Overview*, Supply Chain Council Inc., http://cob.isu.edu/mba691/SCORoverview.pdf

[17] George A. Gecowts and Kenneth B. Ackerman, March/April 2003, *Predictions for the 21st Century*. Supply Chain Management review.

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

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

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

[21] John T. Mentzer,William DeWitt, James S. Keebler, Soonhong Min, Nancy W. Nix, Carlo D. Smith, and Zach G. Zacharia, 2001, "Defining supply chain management," *Journal of Business Logistics*, Vol. 22, No. 2, p. 18.

[22] James T. Lin, Phyllis Chang, Juin-Han Chen, Wei-Xiong Xin, 2001, "KPI with data flow analysis for CPFR: A CMC case study," *International Journal of Business Management*, Vol. 1, No. 3, p. 7.

[23] Colin Adam, "Why Web Services," Webservices.org, 2002, http://www.webservices.org/webs/webs2.html

[24] Scott Bowman, Joe McKinney, and Ray Morgenstern, "Collaborative planning, forecasting and replenishment (CPFR) – White paper and case study," CSC corporation, 2000.

[25] "Super drug and Johnson and Johnson Case study," Syncra Systems Inc, 2001.

[26] "CPFR Technical Specification," Voluntary Interindustry Commerce Standards Association, 1999.

[27] Tamar E. Granor and Doug Hennig, *What's New in Visual FoxPro 8.0*, Hentzenwerke Publishing, 2003

[28] "Getting Started with Liquid Data," BEA Systems, 2004, http://e-docs.bea.com/liquiddata/docs81/qkstart/index.html

[29] "XML – The Big Picture," New Media in Business, 2000, http://www.hyperglossary.co.uk/xml/intro.htm

Appendix - 1

XML Schema and XQuery Code

XML Schema for Scenario 3b:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:scheman XMLns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="resultorderforecast">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="orderforecast" maxOccurs="unbounded">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="forecast" maxOccurs="unbounded">
                        <xsd:complexType>
                           <xsd:sequence>
                              <xsd:element name="itemid" type="xsd:integer"/>
                              <xsd:element name="startdate" type="xsd:date"/>
                              <xsd:element name="enddate" type="xsd:date"/>
                              <xsd:element name="sellerforecastquantity" type="xsd:integer"/>
                              <xsd:element name="buyerforecastquantity"
type="xsd:integer"/>
                              <xsd:element name="hasexception" type="xsd:boolean"/>
                              <xsd:element name="exceptiontypeid" type="xsd:integer"/>
                              <xsd:element name="variation" type="xsd:float"/>
                              <xsd:element name="aggregatedforecast" type="xsd:integer"/>
                           </xsd:sequence>
                        </xsd:complexType>
                     </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="buyerid" type="xsd:integer" use="required"/>
                  <xsd:attribute name="sellerid" type="xsd:integer" use="required"/>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

XQuery Code for Scenario 3b:

```
{--      Bharani Govindasamy - Order Forecast exception generation        --}
let $orderforecast_1 :=
let $forecast_2 :=
for $Orderforecast_Retailer1.forecast_3 in
document("Orderforecast_Retailer1")/orderforecast/forecast
for $Manufacturer.orderforecast_4 in document("Manufacturer")/db/CPFR-
Manufacturer/dbo/orderforecast
for $Orderforecast_Retailer2.forecast_5 in
document("Orderforecast_Retailer2")/orderforecast/forecast
for $AvailableToPromise.Capability_6 in
document("AvailableToPromise")/AvailableToPromise/Capability
let $v_7 := $Orderforecast_Retailer1.forecast_3/quantity -
$Manufacturer.orderforecast_4/quantity
let $cast_as_xs:float_8 := cast as xs:float($v_7)
let $cast_as_xs:float2_9 := cast as xs:float($Manufacturer.orderforecast_4/quantity)
let $div_10 := $cast_as_xs:float_8 div $cast_as_xs:float2_9
let $gt_11 := $div_10 gt 0.1 or $div_10 lt -0.1
let $v_12 := $Orderforecast_Retailer1.forecast_3/quantity +
$Orderforecast_Retailer2.forecast_5/quantity
let $gt2_13 := $v_12 gt $AvailableToPromise.Capability_6/ATPquantity
let $xfext:if_then_else_14 := xfext:if-then-else( treat as xs:boolean($gt_11), 101, 100)
let $xfext:if_then_else4_16 := xfext:if-then-else( treat as xs:boolean($gt_11), 1, 0)
let $xfext:date_from_dateTime_18 := xfext:date-from-
dateTime($Manufacturer.orderforecast_4/startdate)
let $cast_as_xs:integer_19 := cast as xs:integer($AvailableToPromise.Capability_6/itemid)
let $cast_as_xs:date_20 := cast as xs:date($AvailableToPromise.Capability_6/startdate)
where (document("Orderforecast_Retailer1")/orderforecast/buyerid  eq
$Manufacturer.orderforecast_4/buyerid)
and ($Orderforecast_Retailer1.forecast_3/itemid  eq  $Manufacturer.orderforecast_4/itemid)
and ($Orderforecast_Retailer1.forecast_3/startdate  eq  $xfext:date_from_dateTime_18)
and ($Orderforecast_Retailer1.forecast_3/itemid  eq
$Orderforecast_Retailer2.forecast_5/itemid)
and ($Orderforecast_Retailer1.forecast_3/startdate  eq
$Orderforecast_Retailer2.forecast_5/startdate)
and ($Orderforecast_Retailer1.forecast_3/itemid  eq  $cast_as_xs:integer_19)
and ($Orderforecast_Retailer1.forecast_3/startdate  eq  $cast_as_xs:date_20)
return
<forecast>
<itemid>{ xf:data($Orderforecast_Retailer1.forecast_3/itemid) }</itemid>
<startdate>{ xf:data($Orderforecast_Retailer1.forecast_3/startdate) }</startdate>
<enddate>{ xf:data($Orderforecast_Retailer1.forecast_3/enddate) }</enddate>
```

```
<sellerforecastquantity>{ xf:data($Manufacturer.orderforecast_4/quantity)
}</sellerforecastquantity>
<buyerforecastquantity>{ xf:data($Orderforecast_Retailer1.forecast_3/quantity)
}</buyerforecastquantity>
<hasexception>{ xfext:if-then-else( treat as xs:boolean($gt2_13), 1, $xfext:if_then_else4_16)
}</hasexception>
<exceptiontypeid>{ xfext:if-then-else( treat as xs:boolean($gt2_13), 102,
$xfext:if_then_else_14) }</exceptiontypeid>
<variation>{ $div_10 }</variation>
<aggregatedforecast>{ $v_12 }</aggregatedforecast>
</forecast>
where xf:not(xf:empty($forecast_2))
return
<orderforecast buyerid={xf:data(document("Orderforecast_Retailer1")/orderforecast/buyerid )}
sellerid={xf:data(document("Orderforecast_Retailer1")/orderforecast/sellerid )}>
{ $forecast_2 }
</orderforecast>
where xf:not(xf:empty($orderforecast_1))
return
```

Appendix – II

Installation and Use of Liquid Data Software

Liquid data is a software for developing XQuery programs. Liquid data provides an XQuery engine for compiling XQuery code and returning the resulting XML as a Web Service. Let us look into the software requirements for programming and running an XQuery code.

The following software must be installed for using Liquid data as an XQuery engine:
1) BEA weblogic 8.1
2) BEA Liquid Data for weblogic 8.1 SP3

The installable files for these applications can be found in L:/Bharani/Thesis/Software. Install the programs using the default settings suggested during installation.

Liquid data provides a standalone application called "Dataview Builder" for editing XQuery code. Dataview Builder provides a graphical interface to drag and drop the data sources and use the data to form a resultant XML. Hopwever, if the XQuery code is already available, the XQuery source files (.XQ) files must be added to the following folder:

C:\bea\weblogic81\samples\domains\liquiddata\ldrepository

The Subfolders inside the "ldrepository" are to be used for storing the data files and the corresponding XML Schemas. The following are the subfolder and the corresponding files types they store:

delimited_files – Add delimited files such as CSV files here
xml_files – Add XML files here
schemas – This folder contains Schemas for the delimited/xml files and Target XML.
stored_queries – Contains the XQuery programs (its file name is the same as the

web service operation name; i.e., a Web Service operation resultorderforecast() has a corresponding XQuery file named resultsalesforecast.xq.

To add a relational data source, please refer to the BEA Liquiddata documentation at the following link:
http://e-docs.bea.com/liquiddata/docs81/admin/rdbms.html#1052749
As mentioned in the online documentation, the following steps must be done to add a relational data source:

* Creating a JDBC Connection Pool
* Creating a JDBC Data Source
* Creating a Relational Database Data Source Description

After configuring the data sources, we have to start the liquid data server. To start the server, go to Start->Programs->BEA Weblogic Platform8.1->BEA Liquid Data for Weblogic 8.1 SP3->Liquid Data Samples-> Samples Server.
This starts the Samples Server and the Pointbase Server as evidenced by the two windows (console screens) that open up. Wait till the last message on the Samples Server console says, "Server Started : In running Mode."

Note: For any change in the XML Schema, the server has to be restarted to load/reload the new/modified Schemas. However, the server does not have to be restarted to reflect changes in the data available in data sources.

To work with the XQuery files (i.e., to code and test the XQuery), use the Liquid data's Dataview Builder. This gets installed when Liquid data is installed. To open Data View Builder, go to Start->Programs->BEA Weblogic Platform8.1->BEA Liquid Data for Weblogic 8.1 SP3->Data View Builder. For a quick tutorial on using liquid data for data integration, refer to http://e-docs.bea.com/liquiddata/docs81/qkstart/index.html.

Dataview Builder provides functionality to drag and drop operators and functions to develop XQuery code. Care must be taken to choose the appropriate data types which are similar

to the data types used in the XML Schemas. A mismatch of data types will result in a compilation error at Liquid data. The XQuery program that is developed can be tested before using it as a Web Service. Dataview Builder also provides functionality to manually edit the XQuery code.

After the XQuery program is deployed, the XQuery program can be executed as a web service. To convert the XQuery program to a Web Service, we use the admin console of Liquid data. To open the Admin Console, go to Start->Programs->BEA Weblogic Platform8.1->BEA Liquid Data for Weblogic 8.1 SP3->Liquid Data Samples->Admin Console. Use username as system and password as security. Under admin console, select the feature "Run as Web Service." This will make the resulting XML available as a Web Service for use by our Visual FoxPro application.