Islamic University of Gaza

Deanery of Higher Studies

Faculty of Information Technology

Information Technology Program

الجامعـة الإسلاميــة – غـزة

عمــــادة الدراســات العليــا

كلية تكنـولوجيـا المعلومــات

برنامج تكنـولوجيا المعلومـات

Building and Evaluating a SOA-Based Model for Purchase Order Management in E-Commerce System

By:

Yousef M. Al-Ashqar

Supervised By:

Dr. Rebhi S. Baraka

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology

October 2012 / Thul-Qida 1433

الجامعة الإسلامية – غزة
The Islamic University - Gaza

عمادة الدراسات العليا

هاتف داخلي: 1150

الرقم: ج س غ/35/
Ref ....

التاريخ: 2012/10/15
Date ....

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة الدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **يوسف محمود عبد الهادي الأشقر** لنيل درجة الماجستير في كلية **تكنولوجيا المعلومات** برنامج تكنولوجيا المعلومات وموضوعها:

بناء وتقييم نموذج يعتمد على المعمارية الموجهة نحو الخدمة لإدارة طلبيات الشراء في أنظمة التجارة الإلكترونية

Building and Evaluating a SOA-Based Model for Purchase Order Management in E-Commerce System

وبعد المناقشة التي تمت اليوم الاثنين 29 ذو القعدة 1433هـ، الموافق 2012/10/15م الساعة الثانية ظهرًا، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. ربحي سليمان بركة   مشرفاً ورئيساً

د. توفيق سليمان برهوم   مناقشاً داخلياً

د. يوسف نبيل أبو شعبان   مناقشاً خارجيًا

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في كلية **تكنولوجيا المعلومات/ برنامج** تكنولوجيا المعلومات.

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق ،،،

عميد الدراسات العليا

أ.د. فؤاد علي العاجز

Dedication

This thesis is dedicated to the memory of my father Mahmoud Al-Ashqar, my dear mother, my wife, my children (Anas, Almotasembellah, Saja, and Sana) and my brothers and sisters for their patience, encouragement, and support to me during my study of the Master degree.

Acknowledgment

First of all, I thank Allah for giving me the strength and courage to persevere throughout the duration of this research thesis and made all of this and everything else possible.

I am deeply grateful to my supervisor Dr. Rebhi S. Baraka for his continued encouragement, unceasing efforts, persistent motivation, support, and great knowledge throughout my research thesis, without his help, guidance, and follow-up, this research thesis would never have been.

I would like to express my appreciation to the academic staff of information technology program at the Islamic University-Gaza for their help and support me during my master's study and taught me different courses.

I cannot forget to express thanks to my colleagues and classmates for the innumerable amount of help I got from different colleagues and making my study a great experience, useful, enjoyable.

I am greatly indebted to my family for their encouragement and support during my course studies and during my thesis work.

Last but not least, to all named and unnamed helpers and friends, I again extend my thanks.

Abstract

E-Commerce systems are characterized by complex Web applications that use different operating systems and different technologies. One of the most popular E-Commerce applications is conducted between businesses (B2B) and between a business and a consumer (B2C) is Purchase Order Management. It consists of components such as Sales, Shipping and Billing.

In many cases nowadays Purchase Order Management components use integration approaches that lack interoperability and manageability resulting in customer dissatisfaction, time consumption and excessive costs.

In this research, we build a model to overcome shortcomings of current Purchase Order Management system. The model is based on the Service Oriented Architecture (SOA) principles, Enterprise Service Bus (ESB), and Web services. They offer many advantages and help achieve the goals of interoperability and manageability. The proposed model is evaluated by using a scenario based software architecture method and proves that it achieves the quality attributes set as goals for the model which are interoperability and manageability. A case study of the model is implemented as a prove-of-concept. A specific usage scenario for the model is discussed and further proves that the model accomplishes its functionality and quality attributes.

Keywords: *E-Commerce, SOA, ESB, Web services, Interoperability, Manageability, POM.*

عنوان البحث :

بناء وتقييم نموذج يعتمد على المعمارية الموجهة نحو الخدمة لإدارة طلبيات الشراء في أنظمة التجارة الالكترونية

الملخص :

تتميز نظم التجارة الالكترونية بأنها تستخدم مجموعة من التطبيقات الويب المعقدة التي تستخدم أنظمة تشغيل مختلفة وأيضا استخدامها لتكنولوجيات مختلفة.

أحد التطبيقات المفضلة والأكثر استخداماً في بيئات التجارة الكترونية التى تجرى بين الشركات Business-to-Business والشركة والمستهلك Business-to-Consumer هو تطبيقات إدارة طلبيات الشراء Purchase Order Management. حيث أن هذا التطبيق يحتوي على مكونات مثل قسم المبيعات، قسم الشحن، وقسم الفواتير.

كثير من الحالات المستخدمة لتطبيقات إدارة طلبيات الشراء في الوقت الحاضر تستخدم مناهج تكامل تفتقر لصفات التوافقية وقابلية الإدارة بين مكونات أنظمة إدارة طلبيات الشراء التي تؤدي الى عدم رضا الزبائن (العملاء) واهدار الكثير من الوقت وعلاوة على ذلك التكاليف العالية.

في هذا البحث، سنقوم ببناء نموذج model يتغلب على أوجه القصور الموجودة في أنظمة إدارة طلبيات الشراء الحالية. ويستند هذا النموذج على مبادئ المعمارية الموجهة نحو الخدمة Service-Oriented Architecture (SOA)، واستخدام مفهوم ناقل خدمة الأعمال Enterprise Service Bus (ESB)، وخدمات الويب Web Services. وهذه المبادئ والمعايير توفر العديد من المميزات وتساعد في تحقيق أهداف التوافقية وقابلية الادارة. تم تقييم النموذج المقترح من خلال استخدام "منهجية تقييم معمارية البرمجيات بالاعتماد على السناريوهات" Scenario-based Software Architecture Evaluation Methods، للتأكد أن النموذج يحقق الخصائص المطلوبة منه والتي تم تحديدها وهي التوافقية، وقابلية الإدارة. كما تم تنفيذ دراسة حالة (Case Study) ليتم التحقق من اثبات مفهوم النموذج المقترح، وتم فحصها من خلال سيناريواستخدام محدد ومن خلاله تم التأكد من أن المفهوم للنموذج المقترح يحقق الخصائص المطلوبة من حيث الوظائف الرئيسية وصفات الجودة.

Table of Contents

List of Figures

List of Tables

Glossary of Technical Terms and Abbreviations

ANSI   American National Standards Institute (ANSI) is a voluntary organization composed of over 1,300 members (including all the large computer companies) that creates standards for the computer industry.

ASCX12   The Accredited Standards Committee X12, chartered by the American National Standards Institute in 1979, develops and maintains the X12 EDI standards along with XML schemas which drive business processes globally.

ATAM   Architecture Tradeoff Analysis Method is a Scenario Based Software Architecture Evaluation Method

B2B   Business-to-Business, commerce transactions between businesses, such as between a manufacturer and a wholesaler, or between a wholesaler and a retailer.

B2C   Business-to-Consumer, Business that sells products or provides services to end-user consumers.

BC   Binding Components, Container for hosting Service Units that define external connectivity.

BIM   Business Interaction Manager. The field of expertise that encompasses the planning, the design and the control of how a company's business processes interact with the business processes of other internal and external parties.

BPEL   Business Process Execution Language is an OASIS standard executable language for specifying actions within business processes with Web Services. Processes in BPEL export and import information by using Web Services interfaces exclusively

C2C	Consumer-to-Consumer or Citizen-to-Citizen, electronic commerce involves the electronically facilitated transactions between consumers through some third party. A common example is the online auction
DTD	The Document Type Definition (DTD) is a type of document that contains a set of declarations to define the structure of XML files. It is used for XML validation
EAI	Enterprise Application Integration is software and architectural principles that allow for the integration of applications. EAI attempts to provide real-time access to data and processes with minimal changes to the existing applications and their underlying data structures.
EDI	Electronic Data Interchange (EDI) is the structured transmission of data between organizations by electronic means. It is used to transfer electronic documents or business data from one computer system to another computer system.
EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport is the international EDI standard developed under the United Nations. **(EDIFACT)** is an Electronic Data Interchange format used in Business-to-business transactions. It allows EDIFACT message types to be used by XML systems.
EFT	Electronic Funds Transfer is the electronic exchange or transfer of money from one account to another, either within a single financial institution or across multiple institutions, through computer-based systems.
ESB	Enterprise Service Bus (ESB) is a software architecture model used for designing and implementing the interaction and communication between mutually interacting software applications in Service Oriented Architecture.
FTP	File Transfer Protocol (FTP) is a standard network protocol used to transfer files from one host to another host over a TCP-based network, such as the Internet.

G2B   Government-to-Business is the online non-commercial interaction between local and central government and the commercial business sector, with the purpose of providing businesses information and advice on e-business best practices.

G2C   Government-to-Citizen is the communication link between a government and private individuals or residents.

HTML   Hyper Text Markup Language (HTML) is the main markup language for Web pages. HTML elements are the basic building-blocks of Web pages.

HTTP   Hyper Text Transfer Protocol (HTTP) is a networking protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

IETF   Internet Engineering Task Force is the body that defines standard Internet operating protocols such as TCP/IP. The IETF is supervised by the Internet Society Internet Architecture Board (IAB).

J2EE   Java 2 Enterprise Edition (J2EE) is a widely used platform for server
JAVA EE programming in the Java programming language. The Java platform (Enterprise Edition) differs from the Java Standard Edition Platform (Java SE) in that it adds libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server.

JBI   Java Business Integration (JBI) is a specification developed under the Java Community Process (JCP) for an approach to implementing a service-oriented architecture (SOA).

JDBC   Java Database Connectivity (JDBC) is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a

database. JDBC is oriented towards relational databases.

JMS   Java Message Service (JMS) is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients. JMS is a part of the Java Platform, Enterprise Edition. It is a messaging standard that allows application components based on the Java Enterprise Edition (JEE) to create, send, receive, and read messages. It allows the communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous.

JSP   Java Server Pages (JSP) is a Java technology that helps software developers serve dynamically generated Web pages based on HTML, XML, or other document types. JSP was designed to address the perception that the Java programming environment didn't provide developers with enough support for the Web.

OASIS   Organization for the Advancement of Structured Information Standards (OASIS) is a not-for-profit consortium that drives the development, convergence and adoption of open standards for the global information society.

OECD   Organization for Economic Co-operation and Development (OECD) is an international economic organization to stimulate economic progress and world trade.

PIP   Partner Interface Process (PIP) defines business processes between trading partners.

PO   Purchase Order (PO) is a commercial document issued by a buyer to a seller, indicating types, quantities, and agreed prices for products or services the seller will provide to the buyer.

POM   Purchase Order Management (POM) is the process of following a purchase order from inception to delivery of goods or services.

QoS   Quality of Service (QoS) is sometimes used as a quality measure; it refers to the level of quality of service, i.e. the guaranteed service quality.

RPC   Remote Procedure Call (RPC) is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction.

SAAM   Software Architecture Analysis Method is a Scenario Based Software Architecture Evaluation Method.

SAML   Security Assertion Markup Language (SAML) is an XML standard for exchanging authentication and authorization data between security domains.

SEI   The Software Engineering Institute (SEI) is a research, development and training center involved in computer software and network security. The SEI works with industry, academic institutions and the United States government to improve the performance and reliability of computer systems by managing pilot programs, conducting tests, offering courses and providing services for licensing and publication.

SGML   Standard Generalized Markup Language (SGML) is a standard for how to specify a document markup language or tag set.

SMTP   Simple Mail Transfer Protocol (SMTP) is a TCP/IP protocol used in sending and receiving e-mail.

SOA   Service-Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. These services are well-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) that can be reused for different purposes

SOAP   Simple Object Access Protocol (SOAP) is the XML-based protocol used for the message exchange between service users and providers when Web services technology is used.

UDDI   Universal Description Discovery and Integration (UDDI) is a platform-independent, XML based registry for businesses all over the world to be listed on the Internet, UDDI is sponsored by OASIS.

URI   Uniform Resource Identifier (URI) is a string of characters used to identify a name or a resource. Such identification enables interaction with representations of the resource over a network (typically the World Wide Web) using specific protocols. Schemes specifying a concrete syntax and associated protocols define each URI.

VAN   Value-added Network (VAN) is a private network provider (sometimes called a turnkey communications line) that is hired by a company to facilitate Electronic Data Interchange (EDI) or provide other network services.

W3C   World Wide Web Consortium (WWW) was created in 1994 for leading the World Wide Web to its full potential by developing common protocols which promote its evolution and ensure its interoperability.

WSCI   The Web Service Choreography Interface (WSCI) is an XML-based interface description language that describes the flow of messages exchanged by a Web Service participating in choreographed interactions with other services.

WSDL   Web Services Description Language (WSDL) is the XML-based language used to specify service interfaces in the Web services technology. Services are described as a set of endpoints.

WSOA   Web Services Oriented Architectures (WSOA) provide a way to implement scalable Services Oriented Architectures using Web services as elementary services, and orchestrations as composition mechanisms.

XACML   eXtensible Access Control Markup Language (XACML). The standard defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate authorization requests according to the rules defined in policies.

XML   Extensible Markup Language (XML) is a Meta-Language written in Standard Generalized Markup Language (SGML) that allows using to allow for easy interchange of documents on the World Wide Web.

XSD   XML Schema Definition (XSD) is a recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document. This description can be used to verify that each item of content in a document adheres to the description of the element in which the content is to be placed.

Chapter 1

Introduction

This chapter introduces the thesis by describing the E-Commerce systems and the purchase order management, the thesis problem, the research objectives, the importance of the research, the scope and limitations of the thesis work, the methodology, and finally the research format.

Nowadays, E-Commerce systems are characterized by complex Web applications that use different operating systems and technologies and have to face constant changes imposed by technological evolution. E-Commerce sites require the integration of software applications written in different programming languages and residing on different computer hardware distributed across the Internet. This leads to facing issues such as interoperability and manageability.

Purchase order management is one of the applications that are used to purchase goods or services from businesses. Purchasing is the act of buying goods and services that a customer needs. The creation of a management system for purchase orders is often an important part for consumers and businesses [88].

The problem in current Business-to-Consumer/Business-to-Business Purchase Order Management systems is that their constituent components such as sales, shipping and billing use integration approaches that lack interoperability and manageability. This results in customer dissatisfaction, time consumption and excessive costs. This drawback persists strongly when these components belong to different businesses and use different technologies.

Interoperability is the ability of multiple systems with different hardware and software platforms, data structures, and interfaces to exchange data with minimal loss of content and functionality [53]. Interoperability is important because it allows for connecting all parts of purchase order in a uniform way and the exchange of information between parts of purchase order successfully.

Manageability addresses the control and monitoring of applications throughout their lifecycle [64]. Manageability is important since there are different IT systems involved in the purchase order and it leads to achieving proper coordination between different IT sections of purchase order.

XML, Web services, and Service-Oriented Architecture (SOA) are recent standards and technologies that can be used to achieve interoperability and manageability and therefore achieve the required level of integration.

SOA is an architectural and design discipline conceived to achieve the goals of increased interoperability and manageability. It supports the interaction of a variety of services in the system, communicating with each other in unified manner, for realizing integration, information sharing and achieve high interoperability by utilizing the description, discovery, and invocation of services [61-63, 29]. It allows gathering information about services, business processes and supporting tasks such as monitoring, analyzing and resolving faults in services for correctness and consistency between services.

SOA have become a possible component technology in distributed E-Commerce platforms [26, 33, 63]. To be successful, SOA must be implemented on open standards [18, 12].

In this research we propose to use SOA and Web services for building a model for Purchase Order Management in a Business-to-Consumer/Business-to-Business E-Commerce System. To realize the model, we will specify its requirements including interoperability and manageability and base it on suitable standards, approaches and technologies such as Enterprise Service Bus (ESB) [62].

The research would provide enhancement towards the development of Business-to-Consumer and Business-to-Business E-Commerce systems and solving the issues of interoperability and manageability related to the Purchase Order Management leading to reducing cost and efforts of integration and ultimately customers and businesses satisfaction. The importance of the model would allow for easy access, a better connection to the section of Purchase Order in a uniform way and adaptable environment, also it would offer competitive advantages to businesses.

The research is restricted to the domain of E-Commerce especially Business-to-Consumer/Business-to-Business. It examines current development approaches for service integration in order to build the model that follows SOA development strategy to achieve interoperability, and manageability goals.

1.1 Statement of the Problem

In current Business-to-Consumer Purchase Order Management systems, the constituent components such as sales, shipping and billing use integration approaches that lack interoperability and manageability resulting in customer dissatisfaction, time consumption and excessive costs. This drawback persists strongly when these components belong to different businesses and use different technologies.

The problem of this research is how to build a Service Oriented Architecture (SOA) based Purchase Order Management model in Business-to-Consumer/Business-to-Business E-Commerce system that achieves interoperability and manageability.

1.2 Objectives

1.2.1 Main Objective

The main objective of this research is to build a Service Oriented Architecture (SOA) based model for applying Purchase Order Management in Business-to-Consumer/ Business-to-Business E-Commerce system that achieves interoperability and manageability.

1.2.2 Specific Objectives

The specific objectives of this research are:

1. Analyze and determine problems and shortcomings of traditional purchase order management with respect to interoperability and manageability.

2. Specify interoperability and manageability requirements for the SOA based Purchase Order Management model.

3. Build the SOA based Purchase Order Management model suitable for Business-to Consumer E-Commerce system.

4. Realize the model based on current Web services and SOA standards, approaches and technologies.

5. Implement a case study as a prove-of-concept of the model and its realization.

6. Evaluate the proposed model for interoperability, and manageability using scenario based architecture evaluation with ATAM method.

1.3   Importance of the Research

1. The proposed model would have a great value towards the development of Business-to-Consumer and Business-to-Business E-Commerce systems. It contributes in solving the issues of interoperability and manageability related to the Purchase Order Management leading to reducing costs and efforts of integration and ultimately leads to customers and businesses satisfaction.

2. The model would allow for easy access to the section of Purchase Order in a uniform way and adaptable environment and would offer competitive advantages to businesses.

3. The proposed model would allow for a better connection among various sections and facilitates the exchange of information between business and consumer of E-Commerce.

4. Other domains, such as Business-to-Business and E-Government, would benefit from this model by applying similar integration approaches.

1.4   Scope and Limitations

1. This research is limited to the domain of E-Commerce especially Business-to-Consumer/Business-to-Business and the model is proposed for applying Purchase Order based on SOA principles.

2. The scope of the research will examine current development approaches for service integration in order to build the model that have all functionalities and follow SOA strategy to achieve interoperability and manageability goals.

3. An evaluation of the model and its implementation will be conducted through scenario-based architecture evaluation with ATAM method for interoperability and manageability.

4. Security issues are beyond the scope of this research. A limited description will be given for the important usage of security for the services in terms of access control of the model.

5. Performance issues related to efficiency will not be conducted since the model will be implemented on one environment.

1.5   Methodology

To achieve the objectives of the research, the following methodology is followed:

1. Conduct a state of the art review of current E-Commerce systems and related standards and technologies used in E-Commerce systems. Then, Analyze and determine problems and shortcomings of traditional Purchase Order Management related to E-Commerce systems with respect to interoperability and manageability. Additionally, a review of related works and researches will be conducted.

2. Study and investigate the development approaches of SOA for service integration and sharing of information between E-Commerce systems.

3. Investigate possibilities for realizing the model based on SOA and Web services technologies such as Enterprise Service Bus (ESB) to decide which standard would be the most suitable for building the model to achieve interoperability and manageability.

4. Building the proposed model need to realizing and specifying the requirement of system, approaches, services, suitable standards, software and the design, then specifying component of model and their interaction.

5. Implementing a case study as a prove-of-concept of the model and its realization. The perception of the case study will be presenting and Implementing a Bookstore Shop as Business to Consumer E-Commerce system, Then apply and Implementing the section of Purchase Order Management as services based on (SOA) using different standard strategies and Finally, Implement front-end interface that will invoke the implemented services.

6. Evaluate the proposed model using scenario-based architecture evaluation with ATAM method to show that it achieves interoperability and manageability.

1.6   Thesis Format

The thesis consists of seven chapters and is organized as follows:

Chapter One: Introduction

This chapter introduce the thesis by giving short introduction about E-Commerce systems also present the area of the research, thesis problem and research objectives.

Chapter Two: State of The Art and Technical Foundation

This chapter is devoted to introduce and review the concept of E-Commerce and its models. Also this chapter describes the technical foundations needed for this thesis work, like Standards, Principles and technologies such as Web Services, SOA, BPEL, XML Schema and others.

Chapter Three: Related Work

This chapter is to present relevant existing work to this thesis, especially on fields of Business-to-Consumer/Business-to-Business E-Commerce, and researches in the area Service Oriented and Purchase Order Management

Chapter Four: Current Analysis of Purchase Order Management Models

This chapter is devoted to present and analyze the current status POM models, and discusses the issues and problems with it.

Chapter Five: Proposed Model and Case Study Implementation

This chapter is devoted to defining and Present the proposed SOA based POM Model and describes its components and interaction, also present the case study implementation and describe the flow sequence of the model.

Chapter Six: Evaluation of the Proposed Model

This chapter presents the evaluation of the model using scenario based software architecture evaluation method.

Chapter Seven: Conclusions and Future Directions

This chapter presents the final conclusions and future directions.

Chapter 2

State of the Art and Technical Foundation

This chapter introduces the concepts related to E-Commerce and Service Oriented Architecture (SOA). We also review related standards, principles and technologies used in Business-to-Consumer/Business-to-Business E-Commerce Systems are mentioned like Web Services and the Enterprise Service Bus (ESB).

2.1 E-Commerce

E-Commerce has drawn increasing attention recently in areas of research and development in information technology. Numerous papers and reports have been written on E-Commerce and how the concept will change the way companies doing business. However E-Commerce systems are characterized by complex Web applications that use different operating systems and different technologies and have to face constant changes imposed by technological evolution. Integration with existing system become critical issues because a lot of E-Commerce sites require the integration of software application written in different programming languages and residing on different computer hardware distributed across the internet this leads facing issues such interoperability and manageability [88].

E-Commerce is widely discussed and studied, the definition is somewhat arbitrary. Electronic Commerce or E-Commerce has been defined in several ways. E-Commerce is where business transactions take place via telecommunications networks, especially the Internet. E-Commerce describes the process of buying, selling, transforming, or exchanging products, services, and information via computer networks including the Internet [87]. Another description given by [85], E-Commerce is about doing business electronically. The Organization for Economic Co-operation and Development (OECD) provides two types definition for the E-Commerce. One is "electronic transaction" called **broad definition**, including all dealings that make use of computer networks; and the other one is "Internet transaction" called **narrow definition**, referring only to dealings that make use of the Internet technology. The narrow definition is "Internet transaction" including the sale or purchase of goods or services, whether between businesses, households, individuals, governments, and other public or private organizations, conducted over the Internet.

The goods or services are ordered over the Internet, but the payment and the ultimate delivery of the good or service may be conducted on or off-line. This narrow definition excludes orders received or placed by telephone, facsimile or other computer-mediated networks such as Electronic Data Interchange (EDI) [52, 59].

2.1.1 Categories of E-Commerce

There are five distinctive categories of E-Commerce applications namely:

1. **Business-to-Consumer (B2C),** where usually customer can learn about and buy product and services. Customers have opportunity to express their priorities with respect to available products, their delivery, and the way transactions between the customer and the company are to be setup. According to [87], E-Commerce model in which businesses sell to individual shoppers. B2C involves customers gathering information, purchasing physical goods (i.e., tangibles such as books or consumer products) or information goods (or goods of electronic material or digitized content, such as software, or e-books), and for information goods, receiving products over an electronic network. The more common B2C business models are the online retailing companies such as Amazon.com. The more common applications of this type of E-Commerce are in the areas of purchasing products and information, and personal finance management, which pertains to the management of personal investments and finances with the use of online banking tools [4]. Within the B2C category there are many different types of business models. Seven different B2C business models like: portals, online retailers, content providers, transaction brokers, market creators, service providers, and community providers [45].

2. **Business-to-Business (B2B),** is simply defined as E-Commerce between companies. This type deals with relationships between and among businesses. According to [84], B2B is the exchange of products, services or information between business entities. In reference [87], defined B2B E-Commerce as transaction between business conducted electronically over the internet, extranet, intranets, or private network. The major characteristic of B2B is that companies attempt to electronically automate the trading process in order to improve it.

3. **Consumer-to-Consumer (C2C),** E-Commerce model in which consumers sell directly to other consumer [87]. So the E-Commerce involves the electronically facilitated transactions between consumers through some third party. A common

example is the online auction, in which a consumer posts an item for sale and other consumers bid to purchase it; the third party generally charges a flat fee or commission. The sites are only intermediaries, just there to match consumers. They do not have to check quality of the products being offered.

4. **Government-to-Business (G2B),** which emerges as an important policy and implementation instrument for global E-Commerce market and global information society expansion [70]. But Business-to-Government (B2G) E-Commerce can be considered yet another type of E-Commerce, we can subsume B2G E-Commerce within B2B E-Commerce, viewing the government as simply a form of business when it acts as a procurer of goods or services [45].

5. **Government-to-Citizen (G2C),** which comprises the strategic use of the Internet as a channel for the provision of information and services to the citizens in public service sections [70].

2.2   EDI Order Business Model

Electronic Data Interchange (EDI) is described by [24] as the electronic transmission of information or documents between computer systems in different organizations based on a standard, structured, and machine retrievable format. It includes traditional Value Added Network (VAN) which is conducted by private network. According to [32], Electronic Industry Data Exchange gives order business models, through a scenario for traditional stand-alone Purchase Order that is shown in Figure 2.1.
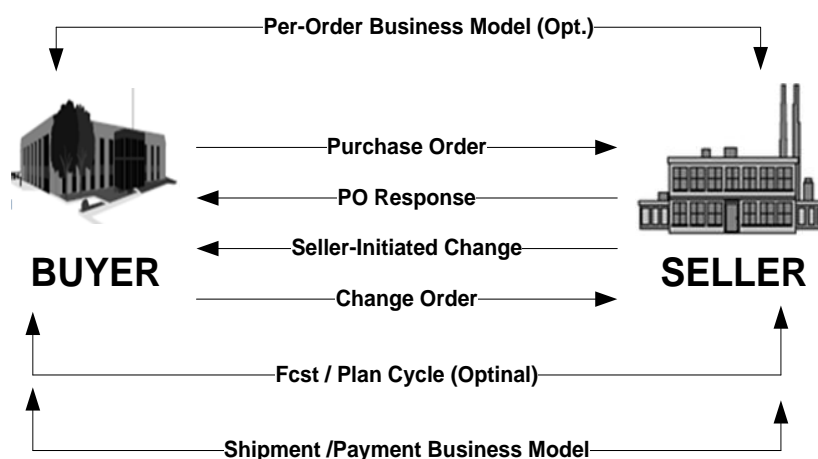
Figure 2.1: Traditional Stand Alone Purchase Order

In the Figure 2.1 the buyer calculates requirement and places purchase order. Based upon agreed standard product and pricing information from price/sales catalog upon

quote request/response. This Model is depends on EDI. So EDI has given the prospect of eliminating paper documents, reducing cost and improving efficiency by exchanging business information in electronic form.

Most of the current EDI-based e-business frameworks are developed based on ASC X12 or EDIFACT standards [31]. These standards mainly provide the frameworks, how to format an EDI message and have a set of components used to define the message format. To exchange EDI-based messages business information between partners shown in Figure 2.2.

EDI Architecture, have four basic steps to perform a single transaction between the buyer (sender) of purchase order to the seller (receiver) of the purchase order:

1) Mapping the data elements in the database for individual transaction type, for example, purchase order.

2) Extraction of the predefined mapped data elements from the database for a specific transaction type, such as, purchase order.

3) Translation of the extracted data in EDI-format, which is now ready for the transmission.

4) Finally, transmit the EDI-formatted data to the receiver network address using the predefined communication protocol.
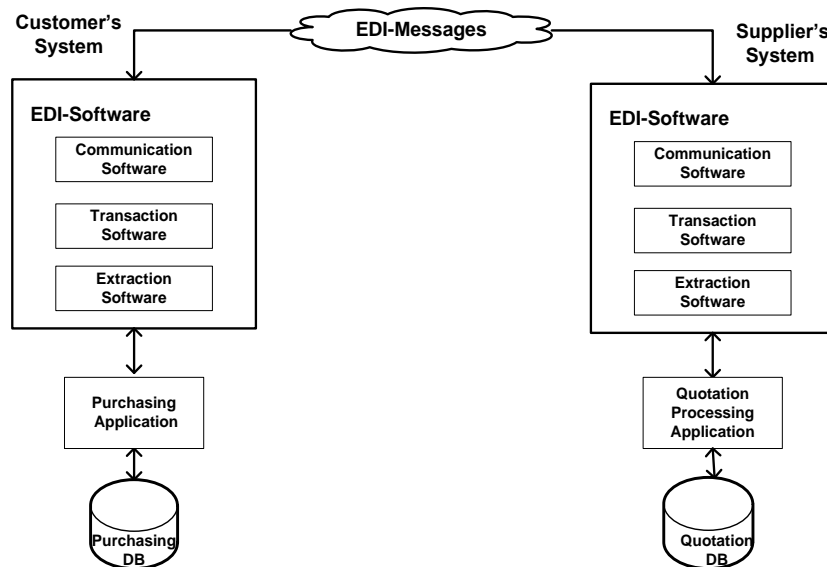
Figure 2.2: EDI Architecture (based on [1])

2.3 XML Effort

A major challenge for E-Commerce is to address the requirements of commercial Web publishing and enable the further expansion of Web technology into new domains of distributed document processing, which has an essential need for a robust, machine-readable information protocol, The World Wide Web Consortium (W3C) has developed Extensible Markup Language XML for applications that require functionality beyond the current Hypertext Markup Language HTML [14]. See the W3C effort [89]. XML is rapidly becoming the standard information protocol for all commercial software such as office tools, purchase order document, messaging, and distributed databases. The W3C has created an Standard Generalized Markup Language (SGML) Working Group to build a set of specifications to make it easy and straight forward to use the beneficial features of SGML on the Web [37]. The goal of the W3C, SGML activity is to enable the delivery of self-describing data structures of arbitrary depth and complexity to applications that require such structures. It is also used to improve the effectiveness of use of large EDI systems.

2.4 Service Oriented Architecture (SOA)

SOA is one of the latest trends and fast moving in computer science industry and very popular with in management due to its practical relevance to business processes. SOA is getting a lot of attention from researchers, so there is many discussions in nearly every professional journal and conference. SOAs are considered as " the next major step in distributed computing " by a big part of the research community today [60].

There are a lot of definitions of SOA, The Organization for the Advancement of Structured Information Standards (OASIS) [58], has published a reference model for SOA, which defines a SOA as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" [58] .

So the technical definition is stated in reference [33]: "SOA is a form of technology architecture that adheres to the principles of service-orientation. When realized through the Web service technology platform, SOA establishes the potential to support and promote these principals throughout the business process and automation domains of an enterprise".

The W3C defines a SOA as "set of components which can be invoked, and whose interface descriptions can be published and discovered".

SOA is a loosely coupled architecture that is based on business requirements rather than any specific technology. SOA is a systems approach to solving the integration problem by building application systems that utilize a common set of business services each of which has been designed and built independent of its various implementations.

The basic SOA is not only architecture about services, but it is also a relationship of three kinds of participants: the service provider, the service discovery agency, and the service requestor. Figure 2.3 shows atypical SOA architecture and the interactions of this figure involve the publish, find and bind operations [13]. SOA uses Web Services as distributed computing technology using protocols such as SOAP, WSDL and UDDI [68], described in next section.
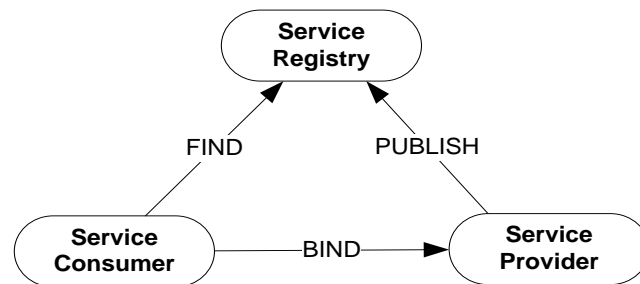
Figure 2.3: A Typical SOA Architecture

2.5   Web services

Web Services rely on a set of standards to support interoperability among applications developed in different languages and run on different platforms or operating systems. Web Services also support the service-oriented approach in SOA enabling interoperability between different systems based on the use of open standards. One way to understand Web services is to understand their standards, including Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), Universal Description Discovery and Integration(UDDI), and emerging Web services flow specifications like Business Process Execution Languages for Web Services (BPEL4WS) and Web Services Chorography Interface (WSCI) [18, 77].

SOAP [23, 18] is a standard for exchanging XML-based messages over a computer network, normally using HTTP. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework, which more abstract layers can build on. This protocol allows exchange of information in a distributed environment and provides a way to communicate between applications running on different operating systems, with different technologies and programming languages. It plays a very important role

in the communication mechanism for Web services. It provides a standard packaging structure for transporting XML documents using a variety of standard internet technologies including SMTP, HTTP, and FTP. A SOAP massage is consists of an "Envelop", an optional "Header" and mandatory "Body". The SOAP body carries actual application information.

WSDL [20, 19, 42] is the standard format for describing a Web service or describing the service interface. The description contains two aspects of the service. Firstly, it has signature and secondly information about binding and deployment details. This information is described using XML. Examples of information are data types used by the service or the address (URI) of the service.

UDDI [42] is an open standard sponsored by OASIS [57], for managing Web Services (i.e., registering and finding services). The aim of UDDI is to provide information about services that can be used by potential users to access Web services. UDDI acts as a directory service that contains service publications. The idea is that services can dynamically locate other services without human intervention. To achieve this, UDDI relies heavily on the WDSL published by the service provider.

In fact, Web services provide an open, interoperable, and highly efficient framework for implementing the Purchase Order Management. They are interoperable because each piece of software communicates with each other piece via the standard SOAP and XML protocols.

2.6 Business Process Execution language (BPEL)

BPEL is a programming language to be written using XML in order to automate business processes. It is provide a composite process from various Web Services and to provide a way to realize processes. BPEL [55] has also known as the (WS-BPEL or BPEL4WS). In fact BPEL have been introduced to address Web service composition issue. WS-BPEL is a specification that models the behavior of Web services in a business process interaction [92]. The concepts WS-BPEL [65, 38] supports two distinct usage scenarios of business processes:

1. Executable business processes model actual behavior of a participant in a business interaction, essentially modeling a private workflow.

2. Abstract business processes specify the public message exchanges between the client and the service.

2.7   Enterprise Service Bus (ESB)

ESB provides a standard-based infrastructure to enable communications between services. It enables high interoperability between distributed systems for services. It makes it easier to distribute business processes over multiple systems using different platforms and technologies [42].The idea of ESB is to provide the means to loosely couple the services and to separate the integration logic from specific application into one manageable place.

ESB would allow services that reside on different platforms and are written with different programming languages to communicate. To do so ESB dose not only provide the means to transport messages from one service to another but also transforms these messages. ESB can be seen as the backbone of any SOA. This means that it also is a logical place to apply, for instance security, policy, accounting and reliability in SOA [62]. ESB is a critical infrastructure that should be implemented and designed based on architectural blueprint. So, an ESB is a product, which evolves from architecture.

An ESB is valuable to the implementation of a service-oriented architecture (SOA) [16] and ESB provides a fundamental support for EAI. It provides a middleware for accessing and transforming information to several protocols such as Java Message Service (JMS), SOAP, HTTP, FTP and TCP. It allows the communication between these different protocols with the support of adaptors [71], where adaptors are used to connect applications to the ESB.

To ensure Interoperability, the components of the ESB and the mechanism for connecting resources must be based on open standard. ESB is realized through using service containers distributed over the network. The containers host integration service such as routers and transformers, and provide services with communication facilities. Messaging infrastructure is built on top middleware systems which guarantee message delivery, such as JMS middleware.

Figure 2.4 depicts a simplified architecture of an ESB that integrates a J2EE application using JMS, a .NET application using a C# client, interfaces with legacy applications, as well as external applications and data sources using web services. An ESB, as portrayed in Figure 2.4 enables the more efficient value-added integration of a number of different application components, by positioning them behind a service-oriented architecture and by applying web services technology. Moreover a distributed query engine is attached to the ESB which is normally based on XQuery or SQL. The

query engine enables the creation of data services to abstract the complexity of underlying data sources. As shown in Figure 2.4, a main use for ESB is to act as the intermediary layer between a portal application server and the backend web services and data sources that the portal application server needs to interact with [62, 64].
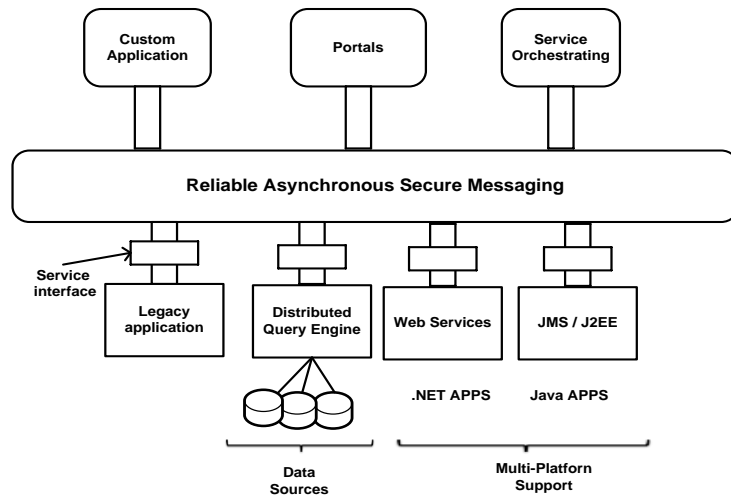
Figure 2.4: A Typical ESB Connecting Diverse applications

2.8   RosettaNet Standards

The RosettaNet [74, 83] has defined a universal e-business standard specific to the electronic and hi-tech industry. RosettaNet is a non-profit organization that provides a standard E-Business language for developing B2B specifications such as supply chain partners in an industry-wide environment. Many of the biggest companies in the world have joined the RosettaNet community in order to make RosettaNet a real powerful E-Business standard.

RosettaNet consortium dedicated to the development of XML-based standard E-Commerce interfaces to align the processes between supply chain partners on a global basis. RosettaNet consists of Partner Interface Processes (PIPs). PIPs are specialized system-to-system XML-based dialogs that define business processes between trading partners. PIPs apply to the core processes of Order Management, Inventory Management and Information Management.

The RosettaNet electronic business model [6] shown in Figure 2.5 is based on supply-chain management architecture where a trading partner is in direct relation with its suppliers and its customers.

For a supply-chain partner to do business in the RosettaNet framework, it must determine its business processes with its suppliers and customers. Partners then define a business scenario that incorporates their supply chain activities into an inter-enterprise business process.
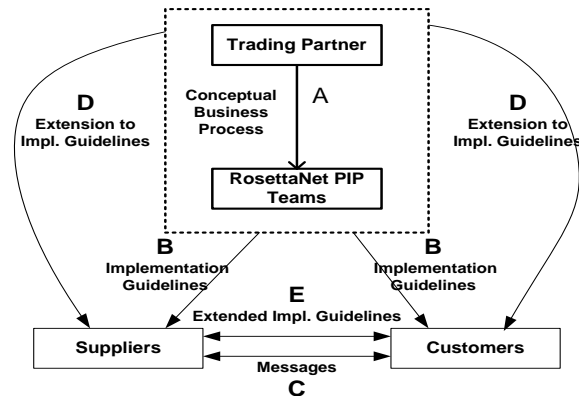
Figure 2.5: RosettaNet E-Business Model

2.8.1 RosettaNet Partner Interface Process (PIP)

A RosettaNet Partner Interface Process (PIP) Models an atomic business process that depicts activities, decisions and partner role interactions that fulfill a business transaction between two partners in a supply chain [73].

A RosettaNet PIPs are a very important part of the RosettaNet B2B process. They are specified XML-based messages, which define all the information needed in a business process between trading partners. The structures of the PIP messages are defined by DTDs (Document Type Definition). A DTD is a specification that accompanies a document, in this case the PIP, and identifies what the code (or markup) in the document means and how it is to be processed. RosettaNet has created Message Guidelines for each PIP to make the structure and contents of a PIP more readable to a human user. A Message Guideline describes exactly the same thing to a human as the DTD to the XML parser [2].

2.8.2 PIP's Functions

- Encapsulate business processes.
- Specifies activities, decisions and roles for the trading partners involved in the transaction.
- Specify structure and format of the business documents.
- Stand as discrete units of work that can be attached and built into other PIPs.

Figure 2.6 show that a PIP-based B2B transaction requires a trading agreement in place between trading partners. In practice, such an agreement is created prior to executing B2B transactions.
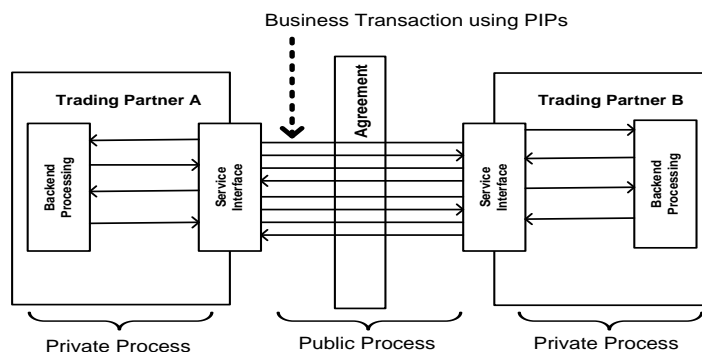
Figure 2.6: Public and Private Processes

RosettaNet has grouped supply chain processes into RosettaNet clusters, which are further grouped into segments. So the PIPs fit into seven clusters, representing the business logic, message flow and message content to enable conversation between partners.

- Cluster 1: Partner, Product and Service Review
- Cluster 2: Product Information
- Cluster 3: Order Management
  - **Segment 3A:** Quote and Order Entry
    - PIP3A1: Request Quote
    - PIP3A2: Query Price & Availability
    - PIP3A3: Transfer Shopping Cart
    - PIP3A4: Request Purchase Order
    - PIP3A5: Query Order Status
    - PIP3A6: Distribute Order Status
    - PIP3A7: Notify of Purchase Order Update
    - PIP3A8: Request Purchase Order Change
    - PIP3A9: Request Purchase Order Cancellation
    - PIP3A10: Notify of Quote Acknowledgement
    - PIP3A11: Notify of Authorization to Build
    - PIP3A12: Notify of Authorization to Ship
    - PIP3A13: Notify of Purchase Order Information
    - PIP3A14: Distribute Planned Order
  - Segment 3B: Transportation and Distribution
  - Segment 3C: Returns and Finance
  - Segment 3D: Product Configuration
- Cluster 4: Inventory Management
- Cluster 5: Marketing Information Management
- Cluster 6: Service and Support
- Cluster 7: Manufacturing

Table 2.1: RosettaNet clusters

Each cluster is broken down into segments, which contain individual PIPs [73]. All the clusters are listed in Table 2.1 as well as all segments in cluster 3 and the PIPs in **Segment 3A**.

We take an example of **PIP3A4**: Request Purchase Order. The Request Purchase Order PIP enables a buyer to issue a purchase order and quickly obtain a response that acknowledges which of the product line items are accepted, rejected or pending. Figure 2.7 describes the business process model for the Request Purchase Order PIP.
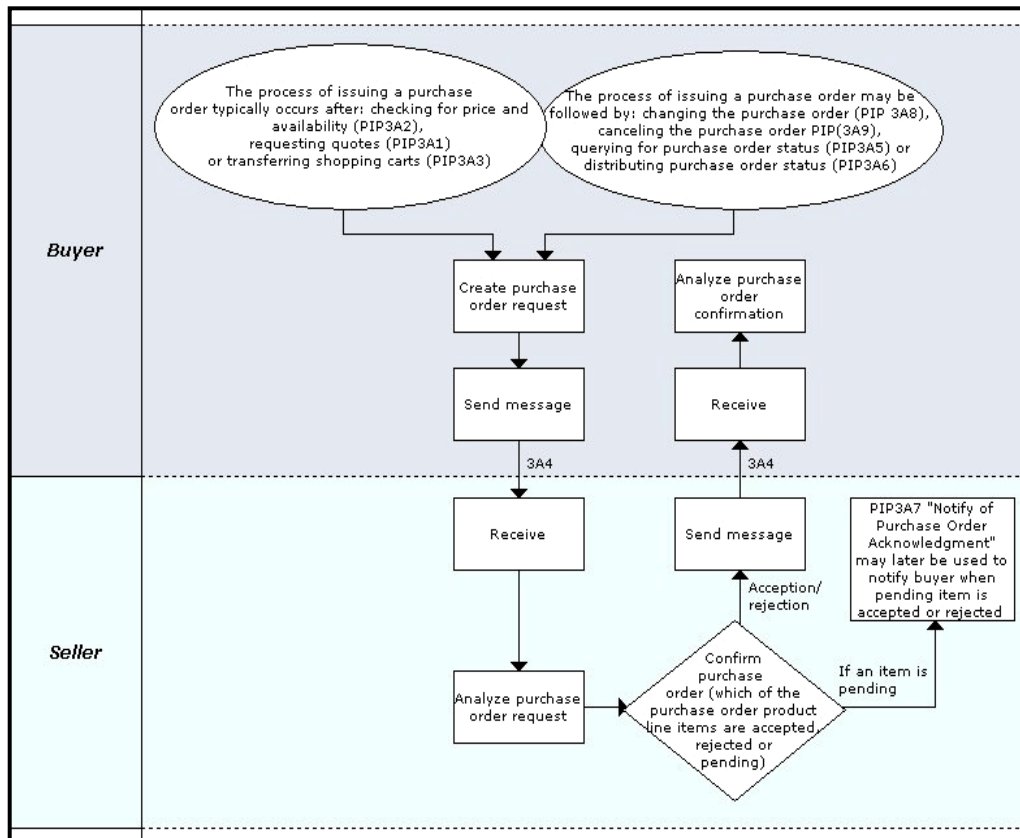
Figure 2.7: Business Process Model of PIP 3A4 [75]

RosettaNet has used a DTD to define the structure of PIPs to using XML schema. PIPs based on DTD, called monolithic, are specified using a considerable amount of free-formatted text, tables, and diagrams that are not machine interpretable. The XML Schema-based PIPs are called modular because they are created with reusable structures. Modular PIPs are specified with considerably more machine-interpretable specifications than PIPs based on DTD.

2.9   Evaluation Strategy

In this section we introduce the evaluation strategy used to test the software architecture and validate the proposed model. Evaluation of application architecture is an important step in any architecture-definition process. The objective of evaluation is to assess whether or not the architecture will lead to the desired quality attributes, interoperability and manageability. Recently, a number of new scenario-based software architecture evaluation methods have been developed by different academic groups and published in the form of books or doctoral dissertations. Many of these methods are refinements of Software Architecture Analysis Method (SAAM) [22] or Architecture Trade off Analysis Method (ATAM) [43, 22].

2.9.1   Architecture Tradeoff Analysis Method (ATAM)

The ATAM [43] approach as it is the method to be used for architecture evaluation. Architectural design decisions determine the ability of system to meet functional and quality attribute requirements. In the architecture evaluation, the architecture should be analyzed to disclose its strength and weaknesses [11].

Two comparison criteria for software architecture are identified, namely, early software architecture evaluation and late software architecture evaluation. In the thesis work we are using the early software architecture evaluation, since we are proposing a model have detailed architecture components. Also it can significantly reduce risks and increase the understanding of the software architecture, verify that all requirements are accounted for get an early indication that quality attribute requirements will be met and an early indication of problems in the architecture. The comparison criteria of early software architecture evaluation has the following features [76]:

- The implementation of oriented data that use scenario-based evaluation method does not need data measured from implementation
- It does not always require metric usage.
- It can be based on mathematical model, simulation based or experience based.
- It requires the participation of the stakeholders.

Both ATAM and SAAM are common methods for architecture evaluation developed by the Carnegie Mellon Software Engineering Institute (SEI) [8]. The ATAM method - relies on the elicitation of quality attribute scenarios from a diverse group of system

stakeholders. ATAM is the most suitable for our proposed model (see chapter 5) since its superior to SAAM and the ATAM is an enhanced method for the SAAM [22]. The purpose of the ATAM is to assess the consequences of architectural decisions in light of quality attribute requirements [43, 22]. Additionally, the evaluation considers a proof of concept method for validate the realization of the proposed model.

The ATAM analysis of the quality attribute scenarios gives insight into how well a particular SOA-based architecture satisfies the particular quality attribute goals of these scenarios and how certain quality attributes interact with each other in an SOA context.

A prerequisite of an evaluation is to have a statement of quality attribute requirements and a specification of the architecture with a clear articulation of the architectural design decisions. However, it is not uncommon for quality attribute requirement specifications and architecture renderings to be vague and ambiguous. Therefore, three of the major goals [43] of ATAM are to.

- Elicit and refine a precise statement of the architecture's driving quality attribute requirements.
- Elicit and refine a precise statement of the architectural design decisions.
- Evaluate the architectural design decisions to determine if they satisfactorily address the quality requirements.

The central goal of an architecture evaluation is to uncover key architectural decisions. The ATAM help to develop a set of analyses, rationale, and guidelines for ongoing decision-making about the architecture. Outputs of the ATAM [43] yields

- A set of risks and non-risks: risks are architecture decisions that might create future problems for some quality attribute requirement. Similarly, a non-risk is an architectural decision that is appropriate in the context of the quality attribute that they affect.
- Sensitivity points: are architecture parameters for which a slight change makes a significant difference in some quality attribute.
- Tradeoffs: are architecture parameters affecting more than one quality attribute.

ATAM method consists of nine steps which are: 1-Present ATAM, 2- Present business drivers, 3- Present the architecture, 4- Identify architectural approaches, 5-Generate quality attributes utility tree, 6-Analyze architectural approaches, 7-Brainstorm and prioritize scenarios, 8-Analyze architectural approaches, 9-Present results.

These steps are typically carried out in two phases. Phase 1 is architect-centric and concentrates on eliciting and analyzing architectural information. This phase includes a small group of technically oriented stakeholders concentrating on steps 1 to 6. Phase 2 is stakeholder-centric, it elicits points of view from a more diverse group of stakeholders, and verifies the results of the first phase. This phase involves a larger group of stakeholders, builds on the work of the first phase, and focuses on Steps 7 through 9 [41].

2.10 Summary

This chapter presented the state of art in E-Commerce. We presented firstly, an introduction of the concept of E-Commerce and related studies, in addition the categories of E-Commerce are discussed. Secondly, the SOA technologies give us the benefits and solution for the thesis problems. Furthermore, the different techniques and concept are presented in this chapter such as ESB, Web services and its associated technologies concept like SOAP, WSDL, and UDDI. RosettaNet Standard is introduced and their function works are discussed. Finally the evaluating strategy is presented. It is used to testify the software architecture and validate the proposed model. ATAM method would be suitable for the evaluation our proposed model.

Chapter 3

Related Work

This chapter presents a review existing work in the field of Business-to-Consumer and Business-to-Business. Subsequently, an existing work and researches in the area Service Oriented and Purchase Order Management will be presented in this chapter.

3.1   Research on Business-to-Consumer (B2C) E-Commerce:

E-Commerce environment, especially Business-to-Consumer had developed rapidly for recent years. It has evolved from Electronic Funds Transfers (EFT), comprising of online shopping and Internet banking, to Electronic Data Interchange (EDI), and comprising companies transfer of documents such as purchase orders or invoices. So that Researchers constantly tried to gain an improved insight into consumer behavior in cyberspace. Along with the development of E-retailing, There is need to easy access of Purchase Order management, researchers continue to explain online shopping behavior from different perspectives [80]. There have been intensive studies of online shopping attitudes and behavior with respect with Consumer, The form of Online Shopping definition (called online buying behavior and Internet shopping/buying behavior) refers to the process of purchasing products or services via the Internet. [48] describes guidelines for designing a good electronic store model, The model divides consumer decision processes into five stages: problem recognition, search for information, evaluation of alternatives, choice, and outcome evaluation. When a problem is recognized, demand for certain products that can eliminate the recognized problem is derived. Product information is collected and alternative products are proposed and evaluated. Once an alternative is chosen, the consumer evaluates the outcome and saves the experience for the future.

Many articles and books describe the business activities typically conducted through E-Commerce, such as purchase order management, order entry, and transaction processing.

Moreover many authors describe the functions found in common E-Commerce applications but do not provide an overall model for Purchase Order Management. For example [87], lists the functionality of some E-Commerce systems like: B2C storefronts, e-procurement, auctions, and enterprise portals. His list for a B2C storefront E-Commerce system concludes that such a system must contain subsystems supporting

an on-line catalog, order transaction processing, and a payment gateway. Also his lists provide some guidance for a functional model, but they do not present a complete picture. In [72], author addresses the challenges to E-business are related to strengthening success factors, establishing barriers to failure, diminishing barriers to success and fighting failure factors. In [25], present some challenges in implementing Business-to-Consumer initiative, One of these challenges is the system integration hurdles: Thus B2C e-business requires various systems, which make up a business as well as those of its Supply Chain Partners, or application Purchase Order Management are integrated. This is because the real power of e-business is when systems are integrated [25].

Purchase Order Management is one of most popular application that conducted between Business-to-Consumers and Business-to-Business. Purchase Order is defined as electronic representation of purchase of goods or services at any stage of its life cycle [91]. The Eastern Michigan University, define Purchasing Order as a commercial document issued by a buyer to a seller, indicating types, quantities, and agreed prices for products or services that the seller will provide to the buyer [30]. Sending a Purchase Order to a supplier constitutes a legal offer to buy products or services.

Purchasing Operation is directly related to Electronic Data Interchange (EDI) [67]. Traditionally, the focus of EDI activity has been on the replacement of pre-defined business forms, such as purchase orders and invoices, with similarly defined electronic forms. The primary goal of EDI was to minimize the cost, effort, and time incurred in paper based business transaction. EDI technology has played a major role in improving various business processes in POM. These business processes require the exchange of business document such purchase order, so the EDI has been developed to help in exchanging these document electronically.

The Conventional Purchase Order Management of direct material can be defined as the process of issuing a Purchase Order and tracking its life cycle until receipt of the ordered material into a buyer's inventory or until the ordered material has been consumed by production. Returns (reverse logistics) and Purchase Order settlement sub processes are also included in the life cycle [78] .

Conventional Purchase Order Management is a business process involving different IT systems. The corresponding sections and their interaction are shown in Figure 3.1. These section depends on different Information Technology (IT) Systems

that built on different components of Legacy Systems, Application Servers, Web server and Database servers [10]. In addition the creation of a management system for purchase orders is often an important part for consumers and businesses.
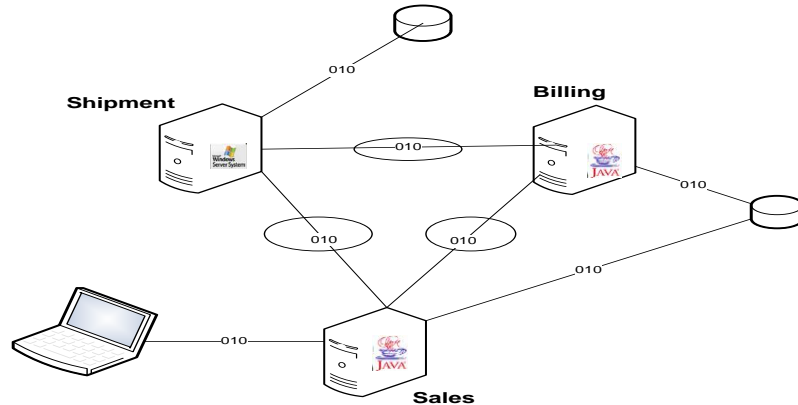
Figure 3.1: Conventional Purchase Order Section

According to [78], the difficulties and lack of visibility between Purchasers and Suppliers of conventional Purchase Order Management is addresses, So that changes to the conventional process order routine are very difficult to assimilate by the buyer and the supplier. Changes may include:

- Modifying a Purchase Order by the buyer (e.g., sending a Purchase Order change request): the supplier must identify and adjust the existing Purchase Order before sending the shipment: the buyer must check the received goods against the modified Purchase Order.

- Partially fulfilling a shipment: partial fulfillment of an order will generate a discrepancy during check-in of the received goods by the buyer. The buyer will then expect a back-order shipment from the supplier.

- A significant amount of time and manual effort is required to monitor the Purchase Order.

Therefore these shortcomings associated with Conventional Purchase Order Management are related to the lack of interoperability and manageability. They appear in the automation of processes (such as sales, shipping and billing) and lead to poor communication and integration between these sections.

Business-to-Consumer and integration model still have several research issues which need to be addressed. Issues associated with Conventional Purchase Order Management can be addressed using SOA. This is due the advantages offered by this type of architecture such as interoperability and manageability. SOA is a flexible and

standardized architecture that allows a better connection among various applications and facilitates the exchange information [42, 34]. The core idea of SOA is the use of services as a distributed computing technology through the use of standard XML protocols [3, 93]. A good effort summary of present research of SOA was presented in [64], defining the Research Roadmap of Service-Oriented Computing with the state of the art of and grand challenges in service research.

According to [63, 64] the directions in the research roadmap of SOA are addressed into 4 layers, which are: service foundation layer, service composition layer, service management layer, and service engineering layer. While the four layers are related to our research topic, the service foundation is the one that is mostly related and should be considered. The state of art for the service foundation layer is to have an infrastructure for Web services and SOA that is realized based on the concept of Enterprise Service Bus (ESB). One of the challenges in this layer is to have an infrastructure that support for data integration, where the infrastructure has the ability to provide consistent access to all the data by all the applications that require it, in whatever form they need it.

To realize the SOA model we must use the Enterprise Service Bus ESB [61, 63]. The ESB is open-standards based message backbone designed to enable the implementation, deployment, and management of SOA-based solutions. The essential ESB requirements that include capabilities such as service orchestration, intelligent routing, provisioning, integrity and security of message as well as service management [62].

In [9], The Central Database model, a core part of the Palestinian e-Government technical framework, is presented and analyzed. A new Central Database model based on SOA solution is proposed that overcomes the shortcomings of the currently used model that lacks the interoperability, flexibility and manageability. The main contribution of this paper is to align SOA conceptual framework to the e-Government domain. Future research efforts can be directed towards the realization of the proposed framework and the challenges that face its realization, as well as to address an enhanced SOA framework that considers governance and policies of Web Services that are published by Central Database as well as local database services providers, both at the central and distributed database levels.

In [35], the main concerns of Enterprise Application Integration (EAI) discusses they are making applications work together and reduce their complexity, and SOA has

been presented as an architectural design style and principles which can better align IT initiative with business requirements. An SOA based enterprise application integration approach is proposed in [35]. Using SOA methods and design styles, the author model the business process of the enterprise and identify process blocks that can be group as services. The existing legacy applications are analyzed to look for functionalities that can be mapped to the identified services. In situations where a service in the business process cannot be mapped to any existing legacy implementation or composed from existing services, the service has to be developed as a new service which is loosely couple, re-useable, extensible as well as interoperable and maintainable. It is verified that this approach ease the composition of existing services and orchestration new business processes. A category of services is defined for the implementation SOA based enterprise applications. They give an example of traveling agency to present how the process services, atomic services and composite services are used to model the business process of the enterprise and identify process blocks using SOA methods and design styles. Figure 3.2 depicts Example of SOA based EAI architecture.
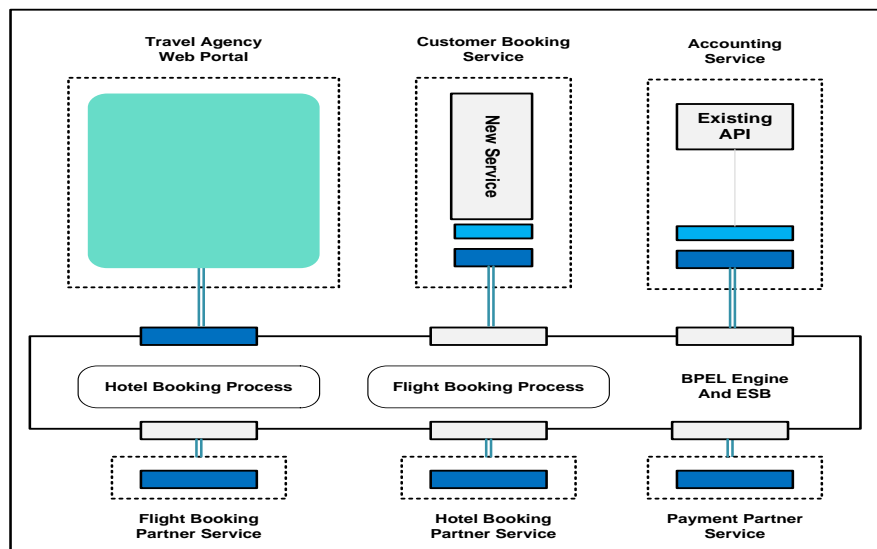
Figure 3.2: Example of SOA based EAI architecture

In [18] authors addressed the implications of Web services that are standardizing the way applications communicate through the World Wide Web. Web services hold the promise to handle and solve complex business problems in the foreseeable future of global competition. They addressed impacts of Web services to E-Commerce research and practices. The research opportunities of Web services and E-Commerce area are fruitful and important for both academics and practitioners.

According to [7] guidance for deploying Web services-oriented architecture (WSOA) enabler of E-Commerce through steady instantiations of the Web services-based business interactions manager (BIM) was proposed. They show that a business interactions perspective, of a business modeling, provides a framework to define the crossing business processes involved in E-Commerce, and to specify the requirements for Web services technology towards Web services based SOA architecture enabler of E-Commerce. They sketched out an approach to steady implement instances of the Web services-based BIM for each category of E-Commerce with respect to the business specifics. This approach is based on a three level architecture where the Web services-based BIM is central to the business processes involved in E-Commerce and the information system. That is, the Web services-based BIM complements and makes the information system a truly backbone of the business, which is a critical for the survival of the businesses willing to cost effectively and dynamically composing external business processes.

A general E-Commerce model based on SOA is presented in [5], the advantages of using a business functionality approach, like obtaining the level of flexibility and interoperability necessary for the systems to adapt to future technological changes. The model takes into consideration the relationship of the company with its suppliers and customers and involves a SOA that takes into consideration the business functionalities of the system. The latest technological changes and the process of globalization turns the economical environment into a collaborative field, where flexible and adaptable informational structures like SOA based models will offer the competitive advantage. Figure 3.3 depicts SOA E-Commerce System Scheme.
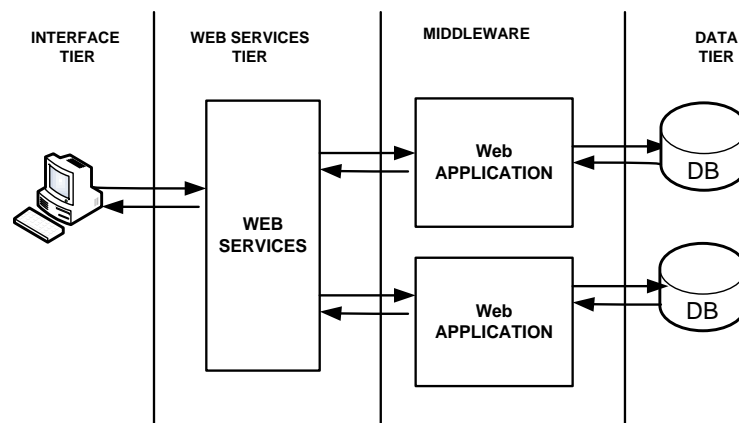
Figure 3.3: SOA E-Commerce System Scheme

According to article in [81] analyzes and compares popular B2B frameworks that attempt to address such issues as interoperability and security between enterprises transacting business over the Internet. In this article standard such RosettaNet that facilitate Business-to-Business integration are discussed.

In [83] they present an infrastructure for B2B exchanges with RosettaNet They have a three-tier client-server prototype that allows customers to send RosettaNet PIPs using a browser. Their prototype constructs RosettaNet PIP service contents. They claim to support all the PIPs and that this kind of solution would mean interoperability. There is no backend integration done and their prototype excludes RosettaNet Implementation Framework (RNIF) functionality and the process aspects of RosettaNet considering times to answer.

In [51], They present a proposed architecture to build solution architectures for B2B E-Commerce hubs platform based on dynamic SOA. It provided a comprehensive business services model and capitalizes on Web services as well as business process management and orchestration. The proposed solution creates a clear, automatic path between the business specification layer and the technical implementation layer by combining both a SOA and management views in a single framework. The paper assessed the capabilities of the proposed architecture in building vertical B2B e-Marketplaces by applying the proposed architecture to the building of a vertical B2B e-marketplace for the oil and gas sector. B2B e-Marketplaces are online hubs that bring together buyers and suppliers to form a trading community. The author state future direction can be extended to enhance the architecture which is decomposition "Dynamic Web Services Definition" of BPEL "Business Process Execution Language" can facilitate efficient SOA in the future, which have not been completely addressed in the scope of his research.

In [40], they provide a brief survey of on-going research in the application of SOA in Business Process Management, examining its usage, its relation with other technologies and related open issues. In particular authors took a specific view of the problem, namely that of the software engineer that is asked to design, develop and implement service architectures under current business process management systems. Author presents a simple case study that captures some of the proposed methodologies and tools and uses such case study to highlight a number of problems related to the implementation of such architectures in real world situations.

In [93], they make concrete analysis and research of the application of SOA in B2B E-Commerce, to integrate the systems of several corporations and use the B2B E-Commerce systems synthetically becomes the most important problem, then propose to use the SOA to build a B2B E-Commerce model based on the J2EE platform, and describe the function of this model logically.

In [49], they address the problem of poor adaptability existing in a shipping E-Commerce platform. They presented a framework based on SOA development to solve the problem of poor adaptability existing in a shipping market. The proposed framework is divided into three layers and provides Web services in three different particle sizes, including application service, BO Service and data persistence service, enhancing the system's reusability.

3.2 Summary

In this chapter a review of existing work in the field of E-Commerce, especially of Business-to-Consumer, Business-to-Business and some work in the field of E-Government was presented. Similarly an existing work and researches in the area of SOA and Purchase Order Management is presented. We found that many authors have described the functions in E-commerce applications but did not provide an overall model for POM. Also there are research directions in the research roadmap of SOA and the contributions of SOA through related technologies. Some research addressed a general E-Commerce model based on SOA and other research presented and analyzed technical frameworks for E-Government and proposes a SOA-based solution.

Chapter 4

Current Analysis of Purchase Order Management Models

In this chapter we analyze the Purchase Order Management (POM) to determine its current status with respect to interoperability and manageability. We specify specific issues with current POM. We analyze the RosettaNet approach to POM. Based on these issues we recommend using SOA-based solution to resolve these issues particularly because interoperability and manageability are central in SOA. We determine most demanding SOA requirement that are needed to achieve the required level of interoperability and manageability.

4.1 Purchase Order Management (POM)

Nowadays, successful Purchase Order Management (POM) is a critical core competency for businesses and customers, since POM have different IT sections and the main concern is about making these sections work together and reducing their complexity. The POM has so many interactions with its sections that must be automated. This requires integration to these sections to automate sharing of contents, communicate and exchanging data between POM sections such purchasing/sales section, inventory section and billing section. These sections are actively facing the interoperability and manageability problems.

The importance of POM to Business-to-Consumer is that facilitates commercial transactions when customers choose and purchase their preferred products. But the Business-to-Business POM is complex to construct and need many requirements such as a huge workflows, synchronization of data and agreement between partners. POM serves a simple purpose that tells some business that a customer wants to purchase some item of goods or services. Actually, a purchase order is legal agreement between business and customer. Because this is a legal document everything needs to be clear as to what is being purchased, where it is to be delivered and what the price is.

4.2 Technical Model of POM E-Commerce System

Traditional POM essentially has four classical main sections: sales, inventory, billing, and shipping and looked as disparate information systems (see Figure 3.1). The technical model of E-Commerce system has different components and layers that need to be realized in order to have a fully functional technical model for the POM system.

Figure 4.1 depicts the POM E-Commerce technical model and its main layers and its different components.
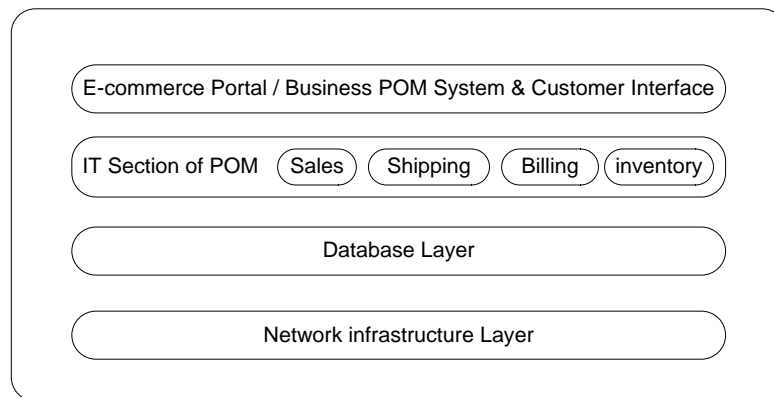
Figure 4.1: POM E-Commerce Technical Model

The IT sections of POM are one of the layers of POM E-Commerce model. The important challenge to the POM model is interaction with these IT sections. It is heavily involve and requiring management consisting of (interoperation, integration and exchanging data between different IT section of POM application).

The POM E-Commerce model is composed of four layers:

1. Portal interface layer:

This layer presents the access interface that customers and businesses interacts with POM E-Commerce system. It is considered as visible part to customers and business to access the Websites and portal and its POM application.

2. IT Section Layer:

This layer contains the core IT section of POM such as sales, inventory, billing, and shipping. This layer provide the portal interface all needed information to exchange between these section to the portal interface layer that is needed to customer and business to complete the process of purchasing items goods or services.

3. Database Layer:

This layer is for storing important data for different section of POM application. This layer address access gateway database sources can easily manage data for IT section of POM such as products information, prices and inventory quantities.

4. Infrastructure Layer:

This layer includes physical and low level software components, telecommunication network, protocols & standards (including EDI, RosettaNet, and XML), operating systems and servers for hosting the database and the application of POM. This layer

presents the interface with networking devices and functionalities such as routing, hosting and control services. Also provides the means to transfer data between network entities.

4.3 Analysis of POM Model

In order to have interoperable and manageable POM E-Commerce model, the IT sections of the technical POM E-Commerce model, discussed in the previous section, must be integrated with each other into single coherent environment. These sections completely isolated from each other are designed to support functionality of POM. This layer has the role of traditional POM depicted in (Figure 3.1). This layer consists of four important applications, namely sales section, inventory section, shipping section and billing, that are cover all process that support POM system. Also these applications have legacy systems such as database application that serve shipment, sales, and billing sections. The databases are managing and querying the data from it. These legacies may support core tasks such as processing orders, generating invoices, credit checking, and inventory checking. This traditional POM model relies on IT sections and synchronization technologies as the low level infrastructure network.

The interoperability between these applications is becoming more challenging because the increased level of connectivity, the data formats becoming more diverse and need for integration to be simple and fast. Interaction between these heterogeneous applications require interoperability at other layers such network infrastructure layer and database layer or other business process. Interacting these applications need to agree on their joint business process such document format the data of the exchanged document as well as the communication protocol to exchange messages. The IT section layer are developed using different programming languages such VB, C, C++, .Net or Java and running on different platforms and using different technologies, and uses components such application servers, Web servers and client browsers to access this traditional model. Managing these components is performed on automation of business processes on using application servers and data servers that span all of these IT sections of POM. The main characteristics of this traditional model are:

- The traditional POM model in B2C E-Commerce processed automatically by computer systems on back ends such server applications and Web browsers.
- The traditional POM model it sections uses some application provided by outsourcing depend on application service provider (ASP) are companies that

package software and infrastructure element together such as shipping or financial systems. ASP is third party that deploys, host, and mange access to package application.

- The POM has different information technology section, these IT section have designed independently as islands of automations with no information integration between them.

- Accessing database is in two way, since the different sections of POM use the databases for updating any changing can be occur or reading the data for checking or validating information entered by customers.

- Customers who have interact with POM using the business Website that offers products which is uses database support. So the Website is based on standard communications protocols of TCP/IP and HTTP and the Web pages are written in the universal recognized standard notions such HTML, and extending the functionality of POM application site, using Java script and ActiveX programs. These tools allow for human interactions, but they still do not address the need to interconnect back end database systems and applications of POM.

- The traditional POM is being done over propriety value added networks (VAN) or over internet.

- The traditional POM is increasing its size and complexity to meet its functional requirements because its different sections use databases, legacy resources and order processing with traditional monitor or management tools.

- Managing and monitoring the traditional POM is performed on traditional tools which cannot automatically monitor and manage the fault that happens on sections of POM in software applications, network connections and hardware resources.

- Traditional POM depends on traditional middleware to connect its sections to translate messages and transactions into specific formats and integrate data flow from sections such sales, billing and inventory that make it difficult to manage these sections.-

- The POM uses Web/client access communication, the business can use Web browser to interact with a Web server for POM sections. Data exchanges, that is extracted from POM application converted into a neutral data format and sent to businesses that use traditional EDI, but the EDI has several limitations and not structured for the internet.

The interoperability and manageability are the major theme in the POM model. It is difficult to design integrated POM model that will work with multiple sections. The interoperability barrier of the POM model makes it difficult to integrate sections to each other with differences of programming languages and technologies of communication protocols which are different from each other in data description and business flow description. These differences make POM model difficult. This is because when any integration section of POM changes its realization mechanism, the other section will have to make changes accordingly. The interoperability is important to POM because the goal of interoperability is to allow different IT sections of POM to work together. This need standards and protocols to describe software or hardware requirements that define common methods for communicating POM sections.

The interoperability standards in general is the description of the message formats exchanged (e.g. purchase order), bindings to transport protocols (e.g. HTTP), the sequencing (e.g. after sending a purchase order message an acknowledgment massage must be received), the process (e.g. after a purchase order is accepted, the goods must be delivered to the customer [15].

The interoperability refers to the ability to share technical and business data, information and knowledge seamlessly across the different IT application of POM.

The ability to capture and share information seamlessly amongst POM sections is very important as it reduce data handling error, facilitates concurrent business activities and improves the responsiveness of an organization. This feature is not always available amongst the currently used POM model. This lack of interoperability is differences to these distributed sections of POM.

Standard that is relevant to POM model such as RosettaNet originally used a distributed approach to creation PIPs [27] (see section 2.8) to facilitate Business-to-Business integration, and support electronic commerce over existing Internet standards and lead to cost and extensibility benefits [81].

The standardized business processes described by RosettaNet simplify cross organizational collaboration and B2B integration. So the degree of standardization offered still requires considerable manual effort during the setup of a B2B collaboration [86]. As a consequence interoperability challenges are only partly addressed by the introduction of standardized PIPs and B2B integrations still suffer from long setup times and high costs. [28] describe the factors that significantly add to the cost of

implementing a PIP. They include ambiguity, incompleteness, and inconsistencies in PIP specifications, lack of reuse across PIP specifications, and the many options in action message contents. These problems require considerable human effort to resolve and increase the cost of implementing PIP-based, multi enterprise collaboration.

The area of B2B integration related to interoperability is still an open issue in E-Commerce systems especially of POM. For example, RosettaNet a PIP3A4, currently used to exchange and Manage Purchase Order to support B2B integration. Has ambiguity inherent in DTD and XML descriptions and incompleteness problem because it has numerous options for elements. Another type of shortcoming that faces PIP3A4 is inconsistency that occurs when the same element consists of different properties, or a different number of properties, across the same or different PIPs.

The scenario presented in [36], details the challenges in a practical B2B integration of RosettaNet collaborations. The scenario presented for purchasing based of RosettaNet PIP3A4 Request Purchase Order. Interoperability issues arise in the following:

- Ambiguous PIP message definitions: partners can use the same PIP messages in different ways. Logically similar information can be represented in many different places, and many elements within a schema definition for a PIP are optional and not implemented by every Business.

- Inconsistences in the naming, syntax and semantics of the elements used in PIPs who interpret the PIP specifications.

- The internal processes according to the global choreography have to be changed for the introduction of every new seller. Not only that multiple messages have to be sent and received, also the decision process to determine which seller is chosen for purchasing has to be introduced.

Other problems when using DTD type to define structure of PIPs described in [27], are specified using considerable amount of free formatted text tables diagrams that are not machine interoperable. There is different terms define the business requirements for each PIP. And each set of business requirements was given to engineers or contactors who often handcrafted individual PIPs without any effort to coordinate the work.

We conclude from the above discussion and mentioned characteristics of the traditional POM model that it is being criticized for various limitations some of them are listed below:

- The use of ASP in some sections of POM model leads to suffering from the inability to manage and integrate the application of POM.

- Incompatibility between syntaxes of the languages of the POM sections.

- It is hard to trace and monitor the flow of transactions within the distributed sections of traditional POM due to the nature of traditional tools, this makes monitoring and management flow difficult. This imposes manageability on POM.

- Using different software operating systems, different software development approaches, different high-level software languages for interfacing data/information, etc. this lack and decrease level of interoperability.

- The standard of PIP3A4 have very little hierarchical information embedded in them it is lacks structure that beneficial when the types of information exchanged change frequently and community agreement processes becomes a bottleneck rather than an aid to interoperability.

- The most common reason is due to incompatibility between the syntaxes of PIP3A4 and the semantics of the terms used by the languages of software application systems. This is mainly due to arbitrary definitions provided by users to the developers of POM application. This will lack the interoperability.

As a result of the above mentioned limitations, there is a strong need for the development of an approach which would overcome these limitations of interoperability and manageability. Such features can be achieved if we adopt SOA solutions and use Web services which promote integration and ensure interoperability and manageability between sections of POM on multiple platforms in seamless communication by using new standards. Web services will provide the standard application interface to support technologies uses in customer and business site to be easier access to the application of POM. However, we only build a SOA-based solution that accomplishes: interoperability and manageability attributes.

4.4   Requirements of SOA-Based Model for Purchase Order Management

Requirements for SOA-Based model for POM need to be defined ahead of presenting the proposed model. To overcome the shortcomings found in current model of POM such RosettaNet, The following requirements must be satisfied:

- The connectivity and accessibility of POM should be based on specific standard connectivity that support interoperability and business need of today that overcomes

the problems. We need compatible standard protocols such as SOAP, WSDL, and XML.

- Business processes requirement. We need standard that achieve collaboration between services of POM in which a primary service directly invoke other services.
- The POM model should include integration to back-end systems of POM and support composition of services.
- Management requirement for the Web services of POM section making it necessary to monitor and manage them for availability. We need suitable and manageable integrations infrastructure for Web services that serve the proposed model.
- The work flow management should be assured for building the model because several activities needed to implement across this requirement that control service invocation and support business processes of POM and support its workflow logic.
- Accessibility and reachability. The POM model should allow the customers easy access through the Internet front end interface.
- Monitoring and Management in POM environment need a new approach such ESB that support monitoring and managing error and fault that occur in services of POM application.

4.5   Summary

POM is analyzed to determine its current status with respect to interoperability and manageability. We specify specific issues with current POM. We analyzed the RosettaNet approach to POM. Based on these issues we recommend using SOA-based solution to resolve these issues particularly because interoperability and manageability are central in SOA concepts. Furthermore we determined the most demanding SOA requirements that are needed to achieve the required level of interoperability and manageability. In the next chapter we develop the proposed SOA based POM model for B2C E-Commerce system.

Chapter 5

Proposed Model and Case Study Implementation

In this chapter we develop a SOA based POM model for B2C E-Commerce system. The model and its components are designed and discussed in detail. The case study of implementation will provide prove of concept of the model.

5.1   SOA-Based POM Model

To realize the requirements discussed in section 4.4 for proposed SOA based POM model for B2C E-Commerce system, we must take into consideration these requirements to achieve the proposed model. The requirements are viewed as different components that achieve relationship between business of services and customer. The proposed model based on Web services and ESB that will allow interoperability with POM section and manageability imposed by using the business processes and ESB that provide methods for monitoring and managing services. The use of Web services in this model allows the separation of POM sections in independent services each with its own functionality and acting simultaneously in order to realize the main functionality of the system. The proposed model is showed in Figure 5.1 in the next page.

The proposed model mainly consists of the following components:

1. The Enterprise Service Bus (interoperable):

This component is considered as one important component of the proposed model it plays the role of integration between different services, and is considered as the middleware support of SOA. It solves problems of interoperability during service module calling to better serve POM application and its section. ESB is a method to manage services. ESB is used and allows customers to access the POM services over the Internet, thus achieve connectivity and accessibility requirement of the model.

2. Service Registry:

The service registry is used to provide a controlled point of access for services metadata for all services provided by POM section. The registry is based on UDDI standard that enable interoperability across heterogeneous services of POM. This component helps in organize and catalog Web services of POM for Sharing and reuse it, also it is provide details on how to publish and locate information about POM Web services.
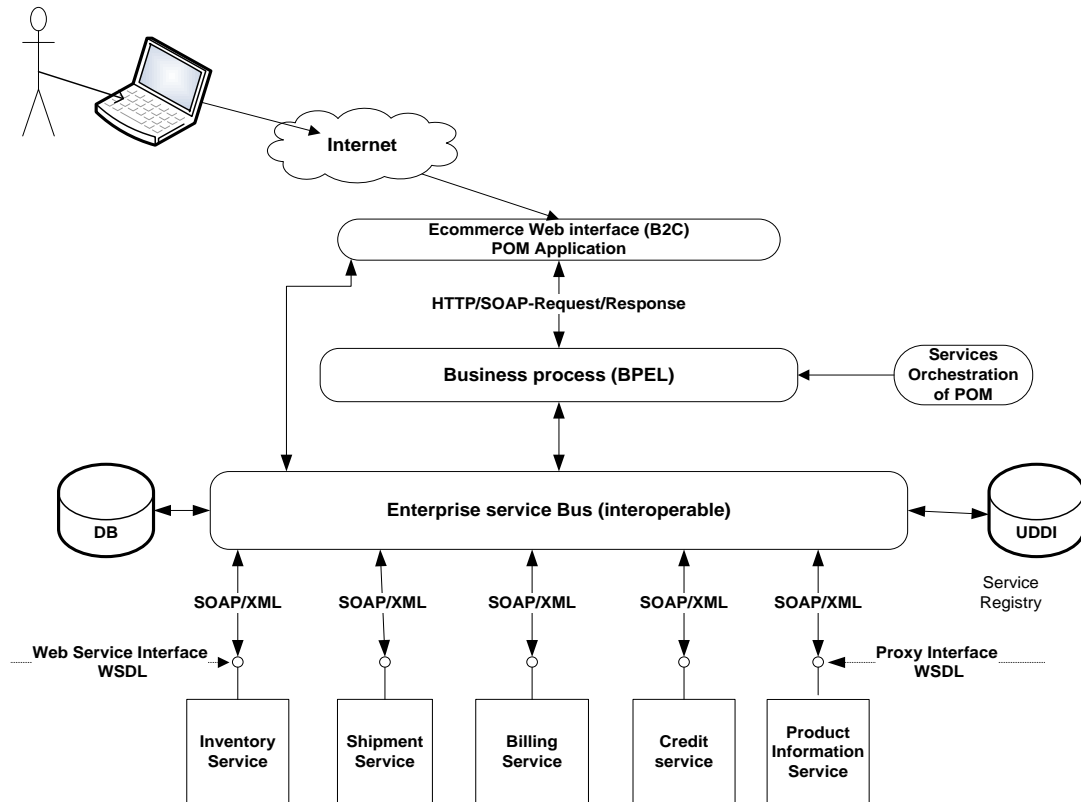
Figure 5.1: Proposed SOA Based Model

3. Database

This component is important for storing the required information need for building the Website or the portal of B2C that consumer interact with. The client requests data about products or services from the portal of B2C to retrieve it. This component achieve accessing and reachability requirement for the model.

4. Core Services of POM

These services include Inventory service, Billing service, Shipment service, Credit service, and Product Information service. These core services realize and manage the POM that configure and serve to order product and provide information about product price and availability. In which Product Information service returns all related information about a product as part of POM. It accesses the required information from the database, the Inventory service is to determine whether there is sufficient inventory of each product to complete a purchase order, the Credit service check and determine whether there is sufficient credits, the shipment service provide the cost of agent may shipping the product over it to specific address of customer, and final service is Billing service generates bills according to approved policies of Credit service and Inventory

service. Invoke these services in composite service using PBEL that interacts directly with the POM which returns status of purchases product.

5. Service Orchestration using BPEL:

This component is important part of POM usually implemented using BPEL, which is responsible for orchestrating business processes or sub business processes and manages composite services. This component is correlated with ESB. In orchestrating, the involved Web services of POM are under control of a single endpoint central process of another service. This process coordinates the execution of different operations of these Web services are participating in the process. The invoked Web services of POM are involved in a composition process and they are playing a role in a business process definition. The composite service is invoked by POM client and it turn it invokes and orchestrate different services of POM. This component will achieve the composition service requirement.

5.2   Interaction of the Model

The interaction between the components of the model is done through the Enterprise Service Bus which integrates the components. It routes, transports, and formats requests and responses of the services, and provides service discovery through the registry.

The proposed model is intended to achieve its goals which are interoperability, and manageability. Interoperability, allows using diverse types of system components of POM like Sales, Shipment, Billing and Product Information. This may have different software, data structure and interfaces to exchange data. Interoperability is important because it allows for connecting all section of purchase order in a uniform way and the exchange of information between sections of purchase order successfully. Interoperability is achieved by having different IT systems, so the automation of the business process by having the several IT systems of the departments interacts.

When services and business processes of the model become operational, their progress needs to be managed and monitored to gain a clear view how services perform their operation. The manageability is achieved by having reporting service execution details, and supporting QoS in Web services of the model such as security. The security requirement has added important because Web service invocation occurs over internet,

it should be considered as manageable for Web service, so it involves aspect such authentication, authorization and confidentiality.

Figure 5.2 is scenario workflow based on the model and shows how the interaction of model occurs to business and customer. This scenario has two sides, the customer side and the business side. The customer side is related to end users that access B2C portal E-Commerce system using a Web browser. The role of the customer is to use POM model for viewing product information and requesting the product of interest in and purchase it by placing an order using POM application and sending the order to the business side and wait for response. If the customer filled a valid order, the product will be sent to the customer with receive total invoice. Otherwise, he will receive error response. On the business side, the POM application as deployed as different services, in which customer can interact with it by browsing product. One of these services is called Product Information service which receives all requests from customer to retrieve the data and information of product from database. The other core service of POM of business side tracking requests when customer placing order by fulfilling order request that realize and mange POM which configure and serve the order request. Core POM services provides response to the customer side of status of Credit Service, and Inventory Service are have sufficient credit, and items. If valid order manipulated in POM business side will send conformation of total invoice to the customer side.
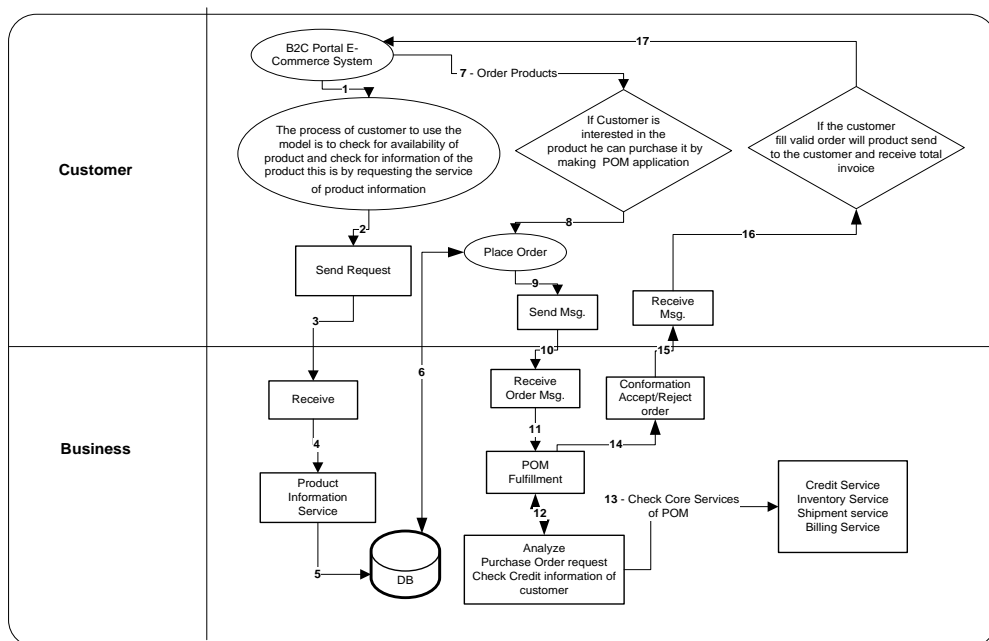
Figure 5.2: Scenario Workflow Based on the Model

5.3   The Security Issue Overview

The security issues are beyond the scope of this research. A limited description is given for the important usage of security for services in terms of access control of the model. We recommend implementing security feature to the model as future work to enhance the model.

Security features is important to POM model since sensitive information is handled in the model such as credit card information and personal information. Security must be assured and should be managed from the logic of the model and accessing all the information through different services of POM. Security polices to be defined and enforced and must not configured at the underling level only, but also at the service level.

The security at service level must achieve the authentication, authorization and logging of accessing different service of the model. Service authentication must be based on identity management using user name and password, and authorization is based on username and IP addressing. The Web client as service consumer must be identified by the user name and password pairs. Access to the service must be allowed if the user is allowed to access the service of the model from the specific IP addresses in which to ensure that clients must access the services from allowed IP addresses.

The strict security requirements are discussed in [66] that consider five security requirement needed to process messages delivered in Web services include authentication, authorization, encryption, integrity, non-repudiation. Also there is several security standards [90, 54, 44] are in development, including Security Assertion Markup Language (SAML), which is a standard for authentication and authorization. Other security standards are XML signature, XML encryption. The ESB handles and satisfies minimum security requirements. In which ESB provides secure messaging to be able to encrypt and decrypt the content of message. Handles authentication and access control for messaging endpoints and to use secure persistence mechanisms.

5.4   Case Study Implementation (Scenario Realization)

To provide a proof of concept of the proposed model, a case study is implemented. In this case study we present a bookstore as B2C E-Commerce system, and through the case study the model is validated to perform its requirements. Briefly we summarize the bookstore as follows:

We suppose in a bookstore to present the products (books) to the customer. The customer wants to access the bookstore using Web browser and purchase some books, the customer contacts with the bookstore and browse information of specific book to order it, the customer in fact contacts with POM application which need to support all the activities of POM section described below:

- Sales section checks with credit card section if credit is okay
- Sales section checks with inventory section if the need books are on store
- Sales section informs the billing section to bill the customer
- Sales section informs the shipment section to send out the books
- Shipment section sends the books to customer
- Shipment section informs the billing section to send the invoice
- Billing section sends the invoice to the customer

5.4.1 High-level of the Case Study Description

Figure 5.3 shows high-level runtime view of the case study. Customer can access the system using a Web browser. The Customer can check the product (book) information such as author name, price, publisher etc.
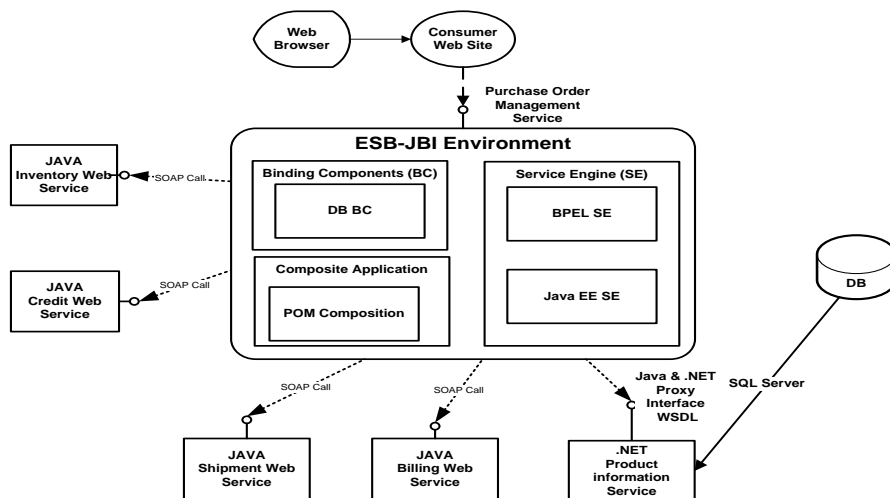
Figure 5.3: Structure the Case Study

After the customer access and view product information, he may place his purchase order. Purchase Order Management tracking request is processed by interacting with the model over Web services interfaces. Formulating a request that handled by Web services. The function available in the front end interface as consumer Web site is access the Purchase Order Management (POM) that interacts with other five services to fulfill order request. The Querying of database will create an answer and send it to the client using SOAP message such XML document as data transport format. The use of

Web services technology allows the separation of independent components, each with its own functionality and acting simultaneously in order to realize the main functionality of the model.

The case study of the model based on two different platforms, namely, Java NetBeans and Microsoft .NET. NetBeans platform runs in context of Java Business Integration (JBI) which is the realization of ESB. Hence: the use of this platform because it was an environment for the management integration component that realize the interoperability and manageability of Web services. The ESB allows connected applications with disparate technology and data formatting requirements to interoperate as service users and providers. The JBI based ESB is provide integration based on an open source standards (OpenESB) is strongly linked with the NetBeans. The JBI is grate enabler for SOA because it defines ESB architecture that contains BPEL Service Engine, HTTP Binding Component, Java EE Service Engine, Data Base Binding, and composite Application for POM Services. Java POM services implemented are four services the Credit Card Service (*CreditWS*), Inventory Service (*InventoryWS*), Shipment Service (*ShipmentWS*), and Billing Service (*BillingWS*). The interaction between Web services over HTTP Binding Component can be used in a composite application. Therefore a single HTTP Binding Component acts as both a service provider and service consumer.

The Java EE Service Engine acts as a bridge between the application server and the JBI runtime environment, facilitating interaction between Java EE and JBI components. The .NET platform it runs in the context of Microsoft ASP.NET applications, the fifth service is Book Information Service (*BookInfoService*) is implemented using .NET with C Sharp language (C#). This service can access information stored in data access layer using SQL Server.

The actors in this Case Study are as follows:

1. Consumer Web Site as (Web Client): Packages up request items in the purchase order as a SOAP request and sends it to the server.

2. HTTP Binding Component: Sends and receives HTTP and SOAP messages between service provider and service consumer.

3. BPEL Service Engine: Responsible for implementing the core business logic of fulfilling the purchase order managements.

4. WSDL Proxy to consume Web services (created in .NET) and services (created in JAVA NetBeans).

5.4.2 The Sequence Processes in the Case Study

For simplicity Figure 5.4 is simple workflow of purchase order Web services and the step performed by POM Web services when client of B2C Web site viewing the product information and request the product. The client may submit a purchase order by placing order using POM application.

1. POM Web service calls credit checking Web service to check the credit worthiness of the client.

2. POM Web service calls the supplier's inventory Web service to check in the items book is available.

3. POM Web service calls the shipping Web service to calculate the shipping cost.

4. POM Web service calls billing Web service to calculate the total bill.

5. POM Web service sends the invoice to the client.
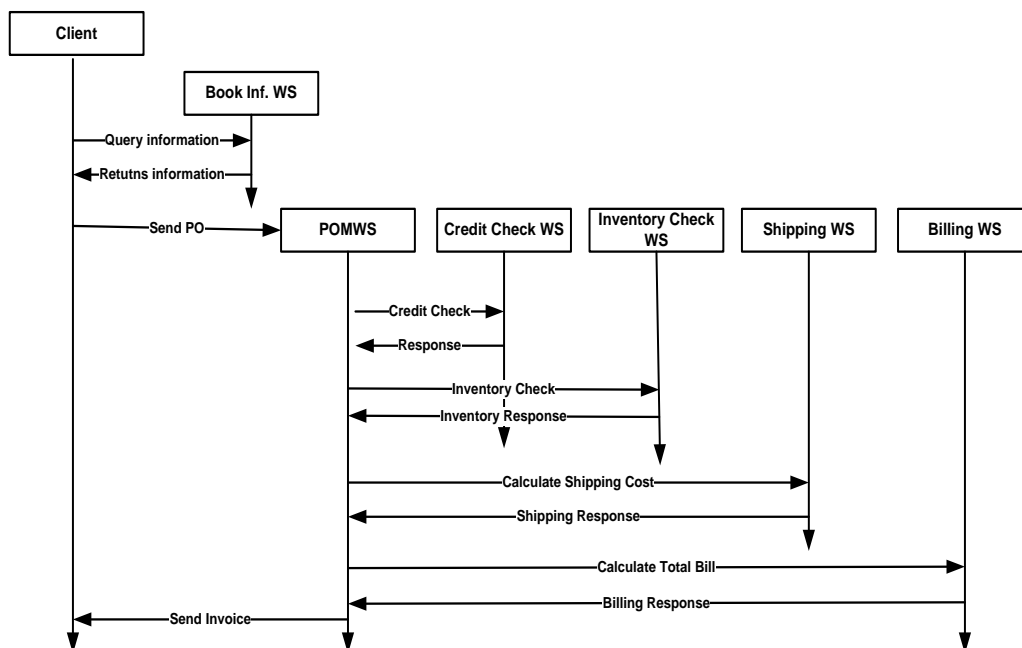
Figure 5.4: Sequence Work Flow of POM Web Services

5.4.3 Web Services Logical View

In this section we present the logical view for the case study Web services. The main function for the model is access the service of purchase order management that invokes other services to fulfill the request of user. The implementation of services is based on top down approach because developing Web services in this way promotes the effort to achieve true interoperability between Web service implementations. We start from all

the required XML schema type definitions XSD, followed by authoring the WSDL document, and then the implementation of the service that interoperates with other Web services. Implementing of services would provide a valuable validation environment to prove the correctness of the case study.

5.4.4   The Core Services of POM

The case study has five Web services: *CreditWS, InventoryWS, BillingWS, ShipmentWS* and one service implemented in ASP.Net name *BookInfoWS* and one composition application.

The Structure of ASP .NET Service *BookInfoWS*: This service provide all related information of books about name of book, author, price, and barcode etc. in detail t this service created as Web service implemented using ASP.NET with C sharp language (C#), we use Data Access Layer (DAL) to access database implemented in Microsoft SQL-Server database. We used DAL from Web Service C# Code. Also we use custom classes in Web service (See Appendix H) these classes go to client in the form of XML Schema so the client can generate class from schema. This service created in ASP .Net can be invoked and consumed from our POM services in java platform, we create proxy WSDL for this Web service in Java NetBeans Platform, and we finally create call method through the proxy to access BookInfo Service Web Service. Figure 5.5 present logical view the .Net Service and interoperable with java NetBeans ESB in the client of POM.
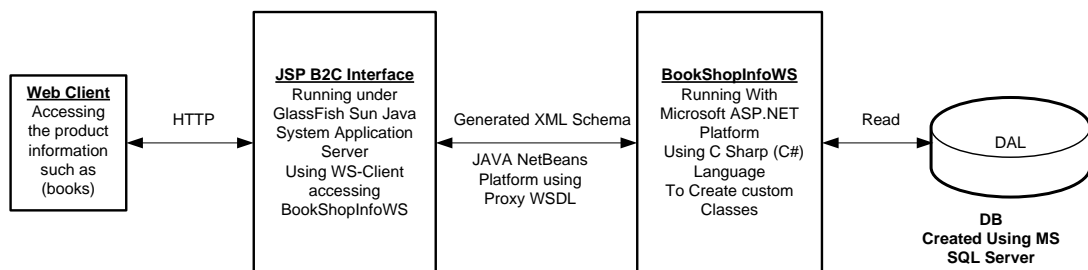
Figure 5.5: ASP.Net Web Service Logical View

The composition applications which orchestrate the four services of POM are implemented in java beans platform which are:

1. The Credit Checking Web Service (*CreditWS*):   This check the validity and worthiness of the credit card. Based on this check, the credit service should respond either that the credit card is valid and worth a credit or by issuing a fault stating that the credit card is invalid or does not have enough credit.

46

2. The Inventory Checking Web Service (*InventoryWS*): This service Provide checks the availability of an item. Based on this check, the inventory service should respond either that the purchase order could be fulfilled or by issuing a fault stating that the order cannot be completed.

3. The Billing Web Service (*BillingWS*): This service should calculate the overall amount and issue the bill to the client.

4. The Shipment Web service (*ShipmentWS*): this service handles the shipment of orders (which are composed of a number of purchase items) to the client.

Figure 5.6 present logical view the core services of Composite POM Web Service
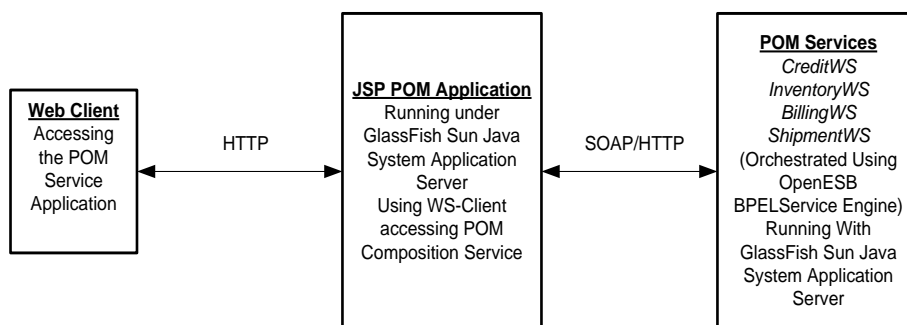
Figure 5.6: Composite POM Web Service Logical View

5. The POM composition is based on the process flow shown in the above Figure: 5.4 is developed by BPEL Service Engine to include credit check Web service, inventory check Web service, billing Web service, and shipment Web service are consumed by BPEL process that realizes the Purchase Order Management Service. The interaction between the Web application accessing the POM Service or the POM BPEL composite application is performed using SOAP over HTTP. The front end or consumer site access the POM service is done using Web browser.

6. When the POM service provider receive the client request of purchased items, the following event occur: the POM service provider assign the price and the current date of the client request, and invoke the Inventory Web Service *InventoryWS* check the availability of an item and report to POM service Provider, Based on the result from Inventory Web Service, the POM Service Provider respond either by fulfilling the POM and request shipment or by issuing fault stating the order cannot be completed. For more information about the implementation of the core POM service and its composition such as XML Schema type definition XSD, Web Service Discovery Language WSDL, and Business Process Execution Language PBEL are detailed in appendix B, C, D, E, F, and G.

5.5 The Requirements Realization of the Proposed Model

To realize the concept of SOA and to achieve the requirements of the model discussed in Section 4.4, the following realizes the requirements of the model.

- The connectivity and accessibility of the model is achieved by using ESB that integrates component of the model of the SOA concept and integrate the services via standards such as XML, SOAP and WSDL. They realize the concept of SOA and to achieve interoperable and manageable integration model for Web services of the POM model.
- The business process is realized and achieved in the model by using service orchestration using BPEL which is responsible for managing composite services of POM which are CreditWS, InventoryWS, ShipmentWS, and BillingWS.
- The composite Services are invoked by the Web client and in turn they invoke and orchestrate the services of POM to achieve the requirement of the composite service.
- Java business integration (JBI) in the model is realized by ESB which integrates back-end systems of POM and supports composite services.
- The work flow management is realized and archived in the model by using service orchestration using BPEL which coordinates the execution of different operations of the services of POM which are CreditWS, InventoryWS, ShipmentWS, and BillingWS which are participating in the business processes.
- Managing and monitoring are realized and achieved in the model using the ESB as manageable and monitoring middleware integration infrastructure for Web services of POM. ESB provides routing, managing and monitoring features for Web services of POM.
- ESB used for accessing the POM services that can be accessed by B2C interface via the internet. This accomplishes accessibility and reachability requirement of the model.
- Accessing the website and B2C interface using different types of databases using SQL query language to retrieve the data and information requested by the Web client. This achieves the accessing and reachability requirement.

5.6 Summary

We proposed the SOA based POM model for B2C E-Commerce system. The model and its component are discussed in detail. We discussed how the interaction occurs between the components of the model. We provide an overview of related security issue for our model, and how security is very important to protect the model. Farther more, we provided a high-level runtime view of the case study it has been realized as a proof of concept for the proposed model and describe its actors. The simple workflow of case study is sketched. Logical view case study Web services are explained. And finally requirements realization is discussed.

Chapter 6

Evaluation of the Model Using the ATAM Method

The aim of this chapter is to present the evaluation and trade off the quality attributes of the proposed model to arrive to a better overall model. The evaluation is conducted against the targeted quality attributes which are interoperability and manageability based on Architecture Tradeoff Analysis Method (ATAM)

6.1 Quality Attributes of the Proposed Model

Software quality attributes are benchmarks that describe the intended behavior of a system within its environment [82]. The quality attribute requirements, such as interoperability, manageability, security, and reliability known as nonfunctional requirements and have a significant influence on the software architecture of a systems [79]. These quality attributes can be specified using quality attribute scenarios. Through this research we focus on two quality attributes to evaluate our proposed model. The quality attributes is clearly defined as follows:

6.1.1 Interoperability

Increased interoperability is the most prominent benefit of SOA, especially when we consider Web services technology [50]. Interoperability is achieved by supporting the protocol and data formats of the service's current and potential clients. Techniques for supporting standard protocol and data formats consist of mapping each platform's characteristics and language to a mediating specification. The mediating specification maps between the formats of the interoperable data format to the platform-specific data formats. Interoperability is also defined as the ability of two or more systems or components to exchange information and to use the information that has been exchanged [39]. With Web services technology and SOA concepts, the following aspect of interoperability has been distinguished [21, 17] Web services as means to reach interoperability between applications. Web services are Web-based applications that provide services through the use of open standards protocols such as XML, Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL). These communication protocols, interfaces, and data formats are the key considerations for interoperability. The use of these open standards boosts and offers a strong interoperability. In SOA concept interoperability distinguished in business, processes, services, and data interoperability [17].

6.1.2 Manageability

From the Web service perspective, a management scheme is needed for maintaining Web service quality. Web services can be managed not only locally by a Web service manager or provider, but also remotely by a consumer. Web services management may be a prerequisite for the foundation of trust between a service consumer and a provider. Manageability is defined as the ability to keep a Web service and its resources being manageable [56]. Also it is defined as a set of capabilities for discovering the existence, availability, performance health, usage, control and configuration of resources within the Web services architecture [69]. The Web services resource includes the software and hardware components used by the Web services and a platform on which the Web services operate. The manageability quality refers to index of ability to consistently manage Web services. Also it can be classified into three sub-factor [46, 47] Informability, Observability, and Controllability. Informability is a sub-quality factor to measure whether the primitive information can provide enough to manage a Web service. Observability is sub-quality factor which measures how effectively a manageability implementation can provide status information of a Web service. The Controllability is sub-quality factor measures whether a manageability implementation can provide enough control functions to keep a Web service in controllable status.

To perform the evaluation we need an evaluation strategy or method to testify the model, as we mentioned in (section 2.9) about evaluation methods that can be used to evaluate software architecture. The evaluation of the proposed model is based on Architecture Tradeoff Analysis Method (ATAM). This is an early evaluation approach for software architecture that is scenario-based. The ATAM focuses on an understanding of the architectural approach that is used to achieve particular quality goals and the implications of that approach. In our case study which is a proof of concept for correctness of the model we will perform a specific usage scenario to show the quality attribute realization.

6.2 Model Evaluation Using the ATAM Method

The ATAM (which is described briefly in section 2.9) analysis of quality attribute scenario gives insight into how well a particular SOA-based architecture satisfies the quality attributes goals of these scenarios and how certain quality attributes interact with each other in an SOA context. The quality attributes for which we are testing the

proposed model for are: interoperability and manageability. We need to set a scenario for these goals and present specific measures for them.

The requirements to conduct the evaluation are evaluation team and stakeholder staff. The evaluation team typically probes the architectural approaches used to address the important quality attributes specified in the scenarios. The goal is to assess wether these quality attribute requirements can be met. In our case the evaluation team is the customer, business administrator and the developer of that uses the systems. Appendix J (The Model Evaluation Scenario Based on ATAM) presents the results of evaluating the quality of the model through three persons represent the three roles.

Figure 6.1 depicts the evaluation process based on ATAM method. The evaluation process relies on evaluation teams, business drivers and constraints, the quality attributes of the model, and architecture approach of the model.
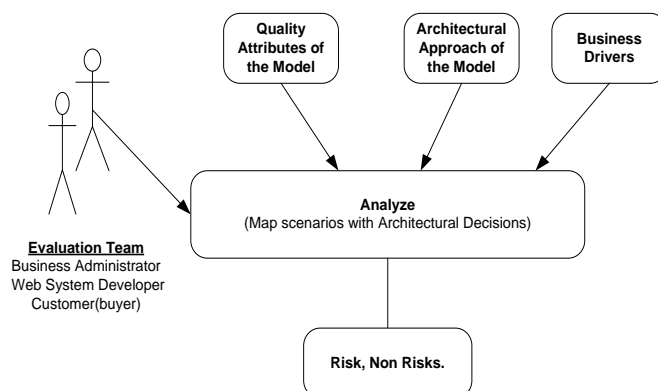
Figure 6.1: Architecture Model Evaluation Based on ATAM

The quality attributes of the model is quality requirements of interoperability and manageability. The architectural approaches of the model are an important architectural information, components used and deployment view of the model. Business drivers is related to business environment, driving requirements, business constrains such standards and customer demands, also related to technical constrains such interoperations with other systems and hardware platforms with related to the major quality attributes goals. The evaluation teams people who will conduct the evaluation such stakeholders, staffs, customers, web developer and administrator. The results of evaluation are scenarios and how far the quality attributes are fulfilled. If the scenario presents a non-risk for the quality attributes the model is considered achieving the quality attribute.

Quality attribute scenario is used to specify quality attribute requirement. In the following section we conduct the qualitative evaluation based on ATAM method. In this

qualitative evaluation we enumerate ATAM scenario under the general scenarios for the important attribute for the proposed model which are interoperability and manageability. The main features that support each of the quality attributes are presented which are inducted from the scenarios. The scenarios are included in the appendix J. Each scenario has a question that concerns the quality attribute and has the prompt that shows the model answers for the question which shows that the model architecture presents a non-risk for the tested quality attributes.

The evaluation questions of the proposed model are given to the specific evaluation team and stakeholders. They are invited to participating in evaluating our proposed model which is depending on the quality attributes of interoperability and manageability. The evaluation team and the stakeholders are three persons and their roles are as customer, network administrator, and web developer of the model. The researcher of this thesis is the developer of the model.

6.3 Interoperability Evaluation Scenario

In this section we present the main features of the proposed model that provide interoperability enhancement. Table 6.1 is a summary for the scenarios (questions and prompts) based on ATAM method that presented in Appendix J.I.

Table 6.1: Interoperability Supporting Features for the Model

Sc.#	Interoperability Supporting Features
1	The proposed model use different services implemented in various platforms and languages.
2	The proposed model is using BPEL for Business Process flow and can orchestrate the difference services by using SOAP and WSDL for service interfacing regardless of the underlying platform or development languages
3	The proposed model allows having service users and providers to use different implementation languages and platforms.
4	The middleware interoperable- integration approach in the proposed model will be an Enterprise Service Bus (ESB). It would allow connecting applications, with disparate technologies, and data formatting.
5	The standards used in the proposed model and provide interoperability between the models components when interacting with each others are: XML, SOAP, WSDL, UDDI, and BPEL which provide capabilities to systems developed with

Sc.#	Web services technology.
Sc.#	**Interoperability Supporting Features**
6	Services may be provided through either synchronous or asynchronous interface on SOA. The selection of service interaction approach depends on the combination of business application logic requirements.
7	A transaction of legacy system such accessing databases running on .Net platform is made available as a Web service in the Enterprise Service Bus (ESB) to integrated with other services running on the NetBeanes Platform. So the ESB provide the interoperability with legacy application.
8	Not all Web services platforms implement the same version of the additional standards such as UDDI, BPEL, WS-Security, and hence achieving interoperability faces some obstacles when using such standards. Since the framework is under a centralized unit of administration, this risk can be mitigated.

6.4 Manageability Evaluation Scenario

In this section we present the main features of the proposed model that provide manageability enhancement. Table 6.2 is a summary for the scenarios (questions and prompts) based on ATAM method that presented in Appendix J.II.

Table 6.2: Manageability Supporting Feature for the Model

Sc.#	Manageability Supporting Features
1	The model using the BPEL that provide and handle the recover fault that can occur in business logic of Web services.
2	The manageability capability as a Web service access endpoint, Also the ESB provide access point of managing Web services.
3	The BPEL engine manages BPEL processes and the application server that runs the engine in its context provides monitoring and logging of event data and measurement of business metrics such as wait time, transaction volumes, and exception counts.
4	Service in the business process can be properly managed and deal with business and technical exception condition when using the composite services in the BPEL workflow

Sc.#	Manageability Supporting Features
5	The model through the ESB and application server provides a monitoring facility for the invoked services, and the health of the framework components
6	The ESB and the application server under which the services run allow starting and stopping the model engines, components, and services.

6.5 Showing Quality Attributes Achievement through a Usage Scenario

In this section we present a usage scenario of the implemented case study of the proposed model. This scenario illustrates how Web services can be used with different systems and the ability of these systems to interact and invoke interoperable services. Figure 6.2 depicts the message flow of this scenario and the interaction direction between different components. The interaction is as follows:

1. The end user – e.g. customer would like to browse the B2C E-Commerce system of Web client as a book shop. This Web client contains books and a customer can view information of any book such as author name or price. In fact the customer browses the client side (Web Client).

2. The Web client, B2C Interface of book shop is using a client-side scripting language like JSP, takes the parameters entered in a SOAP message. The format of SOAP message is defined in the WSDL description.

3. The WS-Client of book information running at client side uses and send SOAP messages over HTTP binding component to invoke .NET book info operation which is part of the BookInfoWS.

4. The .Net service calls the C# implementation methods which are Data Access Layer (DAL) classes to interact with the Database (DB) created using MS SQL Server.

5. The.Net BookInfoWS access the database at the SQL Server.

6. The database returns the requested book record to the implementation method.

7. The implementation method returns the response to the .Net Service BookInfoWS.

8. The .Net service manipulates the result and returns to the SOAP client which returns the result of requested book B2C Interface of book shop.

9. The bookshop store returns information of requested book to the customer.

10. If the customer is interested in the requested book he purchases it.

11. In this case the business of book shop offers the customer a purchase order management application (POM Services) in client side. In this stage Web client, B2C Interface of book shop is using a client-side scripting language like JSP, takes the purchase order information entered into the Web form and packages it into a SOAP message. The format of the SOAP message is defined in the WSDL description.

12. The SOAP message is sent to the POM services endpoint hosted by the HTTP binding component. Then this message is sent to the OpenESB BPEL Service Engine to interpret the purchase order information and properly invokes other BPEL processes to fulfill the request. The OpenESB BPEL Service Engine creates a response message of checking and orchestrating the four services (credit checking, inventory checking, shipment, and billing).

13. The response message sent over HTTP binding component to the POM services and converts it to a SOAP message.

14. The SOAP message is sent back to the Web service client of POM as a proper response as defined by the WSDL.

15. The Web service client takes the response and creates a human-readable HTML page with JSP scripting language to inform the customer whether the purchase order was accepted or rejected. If the order is accepted a will return the Billing of item purchased. If the order is rejected a will return error report due to the checking of credit and inventory error e.g. credit have insufficient amount.
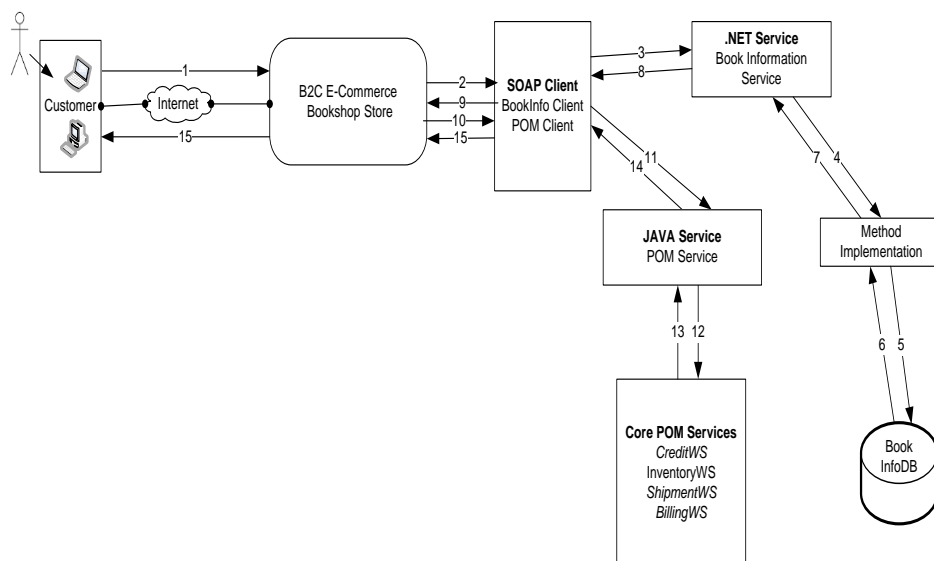
Figure 6.2: Interoperability Usage Scenario for the Proposed Model

Figure 6.3 depicts manageability usage scenario on how to handle run time and business faults, in case of synchronous and asynchronous requests that occur through the model.
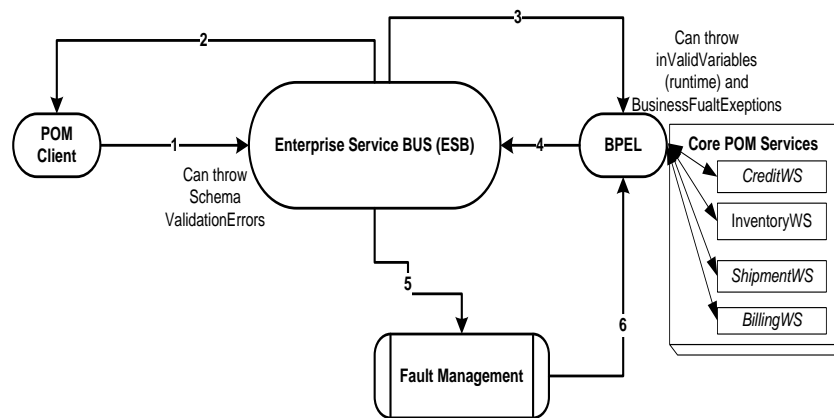
Figure 6.3: Manageability Usage Scenario for the Proposed Model

1. The POM client sends a one-way synchronous request message through ESB.
2. If the ESB detects invalid XML message (schema validation) runtime exception is directly returned to the POM client.
3. The ESB sends a request to BPEL process through routing rule. This is equivalent to stating an asynchronous process
4. BPEL exceptions (both runtime and business faults) are sent to ESB.
5. The exceptions are handled as per polices defined in fault management defined in BPEL.
6. When faults are corrected through retry requests from client, control goes back to the component that raised the exception initially and request completes successfully.

The presented scenarios outline how the implemented case study fulfills the POM to achieve interoperability and manageability quality attributes.

The application of purchase order management described in the scenario is designed to show and verify a certain level of manageability and interoperability between Web services implemented in difference Web services environments.

In these scenarios we connect the parts of POM together with two different environment platforms which are.NET and JAVA NetBeans these different programming languages are interacting with each other via Web service by sending and receiving SOAP

massages formatted in XML over HTTP. The WSDL used to describe the interfaces of the Web service offers a standardized format that fulfills interoperability.

For the quality attribute discussion, the interoperability achievement is clear in the scenario, this because the Web application, .NET service, and java service are heterogeneous services. The scenario setup the Web services based communication between .NET Web services and JAVA NetBeans. The Web service is based on delivered message in scenario between these languages and using database.

The interoperability also is achieved by using standards connectivity such as XML, SOAP, and WSDL that achieve interoperable integration between different services of POM application.

The manageability is achieved by using ESB middleware that manage web services of POM model. It is allowing managing data and transformation of messages between the services of POM model. The ESB provides support for use of propriety orchestration using PBEL which is responsible for managing composite services of POM.

The ESB is responsible for managing and monitoring which using BPEL that provide and handle the recover fault that can occur in business logic of web service of POM model.

Chapter 7

Conclusions and Future Directions

Purchase Order Management model is core part of an E-Commerce environment that is used in B2C and B2B categories of E-Commerce. In this thesis, a Purchase Order Management model was proposed. The model overcomes shortcomings of the currently used model that lacks interoperability and manageability. The proposed model is based on SOA technologies and is realized using ESB and Web services. Web services are adopted as a tool for interoperability. The benefit of Web services for B2C and B2B data integration is one of the reasons why Web services are a relevant technology for the interoperability solution. This research provides the main functionalities of the SOA based POM model and more influential benefits of Web services that use XML and SOAP as the communication bridge between services of POM model. The structure and component of the model were presented and explained. The main components of the model are: Web Services, B2C E-Commerce Web Client, ESB, Business Process of Purchase order Management.

Through the model, we presented how composite services are used to model the business process of POM model and identify the component of SOA. The proposed model realizes the requirements and achieves its goals using Web services and ESB which allow for interoperability and manageability of different services. We achieve the objectives of this research by analyzing the status of traditional POM model with respect to interoperability and manageability features and specifying the requirements of SOA-based model for POM which is related to the connectivity, business processes requirements and managing the work flow requirements and how supporting composition of services. The proposed model realizes all requirements for the proposed SOA-based model using Web services, business processes and ESB middleware. We provide a scenario workflow of the model to show all interactions that occur through the model. A case study implementation as (scenario realization) is to provide a proof of concept of the proposed model. High level runtime view, the sequence processes and the logical view of Web services of the case study are discussed and presented.

In the evaluation of the proposed model, we used ATAM method. The ATAM method is used to evaluate the model in which we introduced different scenarios that affect the

quality attributes of the model. The scenarios were presented using questions and their prompts, which conclude and verify that the model achieves the quality goals. The complete model is not implemented, but limited functions were implemented in the case study. The case study is implemented using different development environments such as Java NetBeans, Microsoft .Net, JSP, JBI, Open-ESB and MySQL database. The implementation includes four services implemented with Java NetBeans environment and one service implemented using Microsoft .Net environments. The orchestration is done using BPEL for the POM services which include four Web services: CreditWS, InventoryWS, BillingWS, and ShippmentWS. The case study is the proof-of-concept to validate the solution of the proposed model and showed that it accomplishes and realizes its requirements.

Table 7.1 summarizes the requirements of the SOA-based model (see section 4.4) for POM model to overcome the shortcoming of traditional model and how they are achieved in our model.

Table 7.1: Summary of requirements and its achievement

#	Requirements	Achieve by
1-	Connectivity and accessibility	Is realized by using standard protocol such as SOAP, WSDL and XML.
2-	Business processes	Is realized through BPEL standard, the Processes in BPEL export and import information by using web service interfaces.
3-	Process Integration	A process can be realized by service compositions, which make ability to schedule the execution of a business process
4-	Managing web services	Is realized through ESB as manageable integrations infrastructure for web services.
5-	Workflow management	The workflow logic is achieved by using Business Process Execution Language (BPEL) for Web Services
6-	Management and monitoring	Is achieved by using ESB and application server provide access point of managing web services and monitoring facility for invoked services.

This research work opens up several interesting directions. To extend this work the integration of business interaction and semantic Web can be adapted and used to enhance the interoperability of the model. The model did not address features that can enhance flexibility and dependability such as service auto-composition. Also the model can be enhanced by adding the security feature. The security feature becomes a major concern on SOA-based Web services. In this research we explored the security issues but did not implement it. Security issues to be addressed should rely on the integration of currently available technologies with evolving security requirements of future Web services of POM application. This requires a unifying technological secure massaging and business policy approaches. Moreover the proposed model did not address the issue of performance. We recommend evaluating the performance as it may affect the services of POM application and the environment that are using the SOA-based model.

References

[1]   Adam, N. R., I. Adiwijaya, et al., "EDI Through a Distributed Information Systems Approach," in *Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences* 1998, pp. 354-363.

[2]   Ajalin, P., U. Briongos, et al., "SAP R/3 Integration to RosettaNet Processes Using Web Service Interfaces," presented at the Introducing BEA WebLogic Integration 8.1, 2004.

[3]   Alonso, G., F. Casati, et al., *Web Services: Concepts, Architectures and Applications*. Berlin: Springer, 2004.

[4]   Andam, Z. R., *E-Commerce and E-Business*, UNDP Asia-Pacific Development Information Programme (UNDP-APDIP) ed.: Wikibooks, 2003.

[5]   Arba, R., "Using SOA for E-Commerce Systems and Development," in *Proceedings of The International Conference on Knowledge Engineering KEPT2009* Cluj-Napoca, (Romania), 2009, pp. 3-6.

[6]   Asuman, D. and C. Ibrahim. (2003). *B2B E-Commerce Technology: Frameworks, Standards and Emerging Issues*.

[7]   Baghdadi, Y., "A Web Services-Based Business Interactions Manager to Support Electronic Commerce Applications," in *Proceedings of the 7th International Conference on Electronic Commerce (ICEC)*, Xian, China 2005, pp. 435-445.

[8]   Bahsoon, R. and W. Emmerich, "Evaluating Software Architectures: Development Stability and Evolution," in *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, Tunis, Tunisia, 2003, pp. 47 - 56.

[9]   Baraka, R. S. and S. M. Madoukh, "A Conceptual SOA-Based Framework for e-Government Central Database," in *Computer, Information and Telecommunication Systems (CITS), 2012 International Conference*, Amman, 2012.

 [10]   Baumeister, H. (2011). *Software Developement of Web Services (Technical University of Denmark ed.)*.

        Available:

         http://www2.imm.dtu.dk/courses/02267/
        http://www2.imm.dtu.dk/courses/02267/index15.html

[11]   Bianco, P., R. Kotermanski, et al., "Evaluating a Service-Oriented Architecture," Software Engineering Institute Technical Report CMU/SEI-2007-TR-015 & ESC-TR-2007-015, 2007.

[12]   Bih, J., "Service Oriented Architecture (SOA) a New Paradigm to Implement Dynamic E-Business Solutions," *ACM Ubiquity,* vol. 7, 2006.

[13]   Booth, D., H. Haas, et al. (2004). *Web Services Architecture* Available: http://www.w3.org/TR/ws-arch/

[14]   Bosak, J., "XML, Java, and the Future of the Web," *World Wide Web Journal* vol. 2, pp. 219-227 1997.

[15]   Bussler, C., "B2B protocol standards and their Role in Semantic B2B Integration Engines " in *Bulletin of the Technical Committee on Data Engineering*, 2001, pp. 3-11.

[16]   Chappell, D., *Enterprise Service Bus Theory in Practice*, - ed.: O'Reilly, 2004.

[17]   Chen, D., "Enterprise Interoperability Framework," in *Proceedings of the Open INTEROP Workshop on Enterprise Modeling and Ontologies for Interoperability (EMOI-Interop)*, Luxembourg, Jan 2006.

[18]   Chen, M., A. N. K. Chen, et al., "Implications and Impacts of Web Services to Electronic Commerce   Research and Practices," *Journal of Electronic Commerce Research,* vol. 4, pp. 128- 139, 2003.

[19]   Chinnici, R., H. Haas, et al., "Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts," presented at the W3C 2007.

[20]   Christensen, E., F. Curbera, et al. (2001). *Web Services Description Language (WSDL) 1.1*. Available: http://www.w3.org/TR/wsdl

[21]   Chung, J.-Y., K.-J. Lin, et al., "Web services computing: advancing software interoperability," *IEEE Computer Society,* vol. 36, pp. 35-37, 2003.

[22]   Clements, P., R. Kazman, et al., *Evaluating Software Architectures: Methods and Case Studies*: Addison Wesley, 2002.

[23]   Clements, T. (2002). *Overview of SOAP*.

 Available: http://developers.sun.com/appserver/reference/techart/overview_soap.html

[24]   Copeland, K. W. and C. J. Hwang, "Electronic Data Interchange: Concepts and Effects," in *Internet Society's seventh annual conference, INET'97*, Kuala Lumpur- Malaysia, 1997.

[25]   D'Costa, M. (2008). *E-Business & E-Commerce Concepts*.

        Available: http://www.scribd.com/doc/31472948/e-Biz-Notes

[26]   Daconta, M. C., L. J. Obrst, et al., *The Semantic Web - a Guide to The Future of XML, Web Services and Knowledge Management*: Wiley, 2003.

[27]   Damodaran, S., "B2B integration over the Internet with XML: RosettaNet successes and challenges," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* New York, USA, 2004.

[28]   Damodaran, S., "RosettaNet: Adoption Brings New Problems, New Solutions," in *Proceedings of the XML 2005 Conference*, Atlanta, USA, 2005, pp. 1-14.

[29]   Dongsheng, Z. and M. Wenshan, "Research and Design E-procurement System of NH Group Based on SOA," presented at the Conference on Industrial and Information Systems (IIS) 2010, Dalian, China, 2010

[30]   Eastern-Michigan-University. (Revised June 27, 2011, Purchasing Procedures. Available: http://www.emich.edu/purchasing/docs/EMU%20PurchasingProcedure.pdf

[31]   EDI-Guide-Home. *History of Electronic Data Interchange*.

        Available: http://www.edi-guide.com/edi-history.htm

[32]   EDIX. (1996). *Order Business Models*.

        Available: http://www.ti.com/sc/docs/scedi/ordmodel.pdf

[33]   Erl, T., *Service-Oriented Architecture: Concepts, Technology & Design*: Prentice Hall, 2005.

[34]   Fujitsu. (2008). *Real-World SOA: Definition, Implementation and Use of SOA with CentraSite™*.

        Available: http://www.webservices.org/recommended/real_world_soa_definition_implementation_and_use_of_soa_with_centrasite

[35]   Gu, C. and X. Zhang, "An SOA Based Enterprise Application Integration Approach," in *Proceedings of the 2010 Third International Symposium on Electronic Commerce and Security (ISECS)*, Guangzhou, 2010, pp. 324 - 327

[36]   Haller, A., P. Kotinurmi, et al., "Handling Hetrogeneity In RosettaNet Messages," in *Proceedings of the 22nd Annual ACM Symposium on Applied Computing (SAC)*, Seoul, Korea, 2007.

[37]   Hammond, W. F., "Extensible Markup Language (XML) Standard Generalized Markup Language (SGML)," 2006.

[38]   Hmida, M. M. B., C. Boutrous-Saab, et al., "Dynamically Adapting Clients to Web Services Changing," in *Proceedings Workshop on Emerging Web Services Technology (WEWST)* Zurich, 2006, pp. 91-96.

[39]   IEEE, *IEEE standard computer dictionary: A Compilation of IEEE Standard Computer Glossaries,* : Inst of Elect & Electronic, , 1991.

[40]   Ivanyukovich, A. and M. Marchese, "Service Oriented Architectures for Business Process Management Systems " in *International Association for Development of The Information Society (IADIS)*, 2005, pp. 543-550.

[41]   Jones, L. G. and A. J. Lattanze, "Using the Architecture Tradeoff Analysis Method to Evaluate a Wargame Simulation System," Software Engineering Institute, Report Number CMU/SEI-2001-TN-022, 2001.

[42]   Josuttis, N. M. (2007). *SOA in Practice: The Art of Distributed System Design*. Available: http://www.soa-in-practice.com/

[43] Kazman, R., M. H. Klein, et al., "ATAM: Method for architecture evaluation," Software Engineering Institute, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2000-TR-004, 2000

[44] Kim, Y. J., "Access Control Service Oriented Architecture Security," Washington Universityin St. Louis, Student Reports project on Recent Advances in Network Security, 2009.

[45] Laudon, K. and C. G. Traver, *E-commerce: business, technology, society*: Prentice Hall, 2004.

[46] Lee, Y., "Interoperability management for SOA by relative proof and test suit," in *6th International Conference on Advanced Information Management and Service (IMS)*, Seoul, 2010, pp. 515 - 520.

[47] Lee, Y. and U. Lee, "QAM: QoS-assured Management for Interoperability and Manageability for SOA," in *5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*, 2010, pp. 88 - 92

[48] Liang, T. P. and H. J. Lai, "Electronic Store Design and Consumer Choice: An Empirical Study," in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Hawaii 2000.

[49] Lin, C. and J. Yao, "Research on Shipping E-Commerce Platform Based on J2EE and SOA Development Technique," in *Proceedings of 7th International Conference on Service Systems and Service Management (ICSSSM)*

Tokyo, 2010 pp. 1–4.

[50] McGovern, J., S. Tyagi, et al., *Java Web Service Architecture, Chapter 2*. San Francisco: Morgan Kaufmann Publishers, July, 2003

[51] Mohamed, U. A., G. H. Galal-Edeen, et al., "Building Integrated Oil and Gas B2B E-Commerce Hub Architecture Based on SOA," in *Proceedings of International Conference on E-Education, E-Business, E-Management and E-Learning (IC4E)*, 2010, pp. 599-608.

[52] Motohashi, K., "Measuring E-Commerce in Japan: Statistical Issues and Challenges," in *International Conference on Information Statistics of the Internet: Measurement, Analysis and Applications*, Macao, Hong Kong, 2004, pp. 179-191.

[53] NISO, " Understanding Metadata," ed: NISO Press, 2004

[54] OASIS-Standard. (2004). *Web Services Security (WS-Security)*.

Available: http://saml.xml.org/ws-security-oasis-standard ,

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[55] OASIS-Standard. (2007). *Web Services Business Process Execution Language Version 2.0*.

Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[56]   OASIS. (22 July 2011). *Web Services Quality Factors Version 1.0*

Available: http://docs.oasis-open.org/wsqm/wsqf/v1.0/WS-Quality-Factors.html

[57]   OASIS. (2004). *UDDI Spec Technical Committee Draft*.

Available: http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm

[58]   OASIS. (2006). *Reference Model for Service Oriented Architecture*.

Available: http\\www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[59]   OECD, *OECD Guide to Measuring the Information Society 2011*. Paris: OECD Publishing, 2011.

[60]   Papazoglou, M., "Service -Oriented Computing: Concepts, Characteristics and Directions," in *International Conference on Web Information Systems Engineering (WISE)*, 2003.

[61]   Papazoglou, M., "Web Services Technologies and Standards," *Submitted to ACM Computing Surveys,* 2006.

[62]   Papazoglou, M. and W. J. Heuvel, "Service Oriented Architectures: Approaches, Technologies and Research Issues," *VLDB Journal* vol. 16, pp. 389-415, 2007.

[63]   Papazoglou, M., P. Traverso, et al., "Service-Oriented Computing: State of The Art and Research Challenges," *IEEE Computer* vol. 40, pp. 38-45, 2007.

[64]   Papazoglou, M., P. Traverso, et al., "Service-Oriented Computing Research Roadmap," *International Journal of Cooperative Information Systems,* vol. 17, pp. 223-255, 2008.

[65]   Peltz, C. (2003, Web Service Orchestration: a review of emerging technologies, tools, and standards.

Available: http://xml.coverpages.org/HP-WSOrchestration.pdf

[66]   Peterson, G. (2008) Security in SOA - It's the Car, Not the Garage *SOA Magazine*.

Available: http://soamag.com/I15/0208-2.pdf

[67]   Petravick, S. (Spring 2002). *Introduction to EDI-A Primer*.

Available:

http://bradley.bradley.edu/~simonp/atg383/
http://bradley.bradley.edu/~simonp/atg383/edipindx.html

[68]   Petritsch, H. (2006). *Service-Oriented Architecture (SOA) vs. Component Based Architecture*.

Available: http://petritsch.co.at/download/SOA_vs_component_based.pdf

[69]   Potts, M., I. Sedukhin, et al. (2003, Web Services Manageability – Concepts.

Available:

> http://www.oasis-
> open.org/committees/download.php/3470/Web%20Service%20Manageability%
> 20-%20Concepts%201.0.pdf

[70]   Poulymenakou, A. and L. Tsironis, "Quality and electronic commerce: a
partnership for growth," *The TQM Journal,* vol. 15, pp. 137 - 151, 2003.

[71]   Rademakers, T. and J. Dirksen, *Open Source ESB in Action, Example
Implemntations in Mule and ServiceMIX*: Manning Publisher, 2008.

[72]   Reiss, M., "E-business: basics and challenges," in *In Proceedings
Photogrammetric Week 2001*, Stuttgart, 2001, pp. 315-328.

[73]   RosettaNet. *PIPs*.

> Available:
> http://www.rosettanet.org/Standards/RosettaNetStandards/PIPs/tabid/475/Defaul
> t.aspx

[74]   RosettaNet. *The RosettaNet Background Information*.

> Available: http://www.rosettanet.org/AbouttheStandard/tabid/276/Default.aspx

[75]   RosettaNet. *RosettaNet PIP3A4*.

> Available:
> http://www.rosettanet.org/dnn_rose/DocumentLibrary/tabid/2979/DMXModule/
> 624/Command/Core_ViewDetails/Default.aspx?EntryId=9574

[76]   Roy, B. and T. C. N. Graham, "Methods for Evaluating Software Architecture:
A Survey," School of Computing, Queen's University at Kingston, Ontario,
Canada Technical Report No. 2008-545, 2008.

[77]   Ryman, A. (2003). *Understanding Web Services*.

> Available:
> http://www.ibm.com/developerworks/websphere/library/techarticles/0307_ryma
> n/ryman.html

[78]   Sabri, E. H., A. P. Gupta, et al. (2006). *Purchase Order Management Best
Practices: Process, Technology, and Change Management*.

> Available: http://www.openisbn.com/isbn/1932159630/

[79]   SEI-Software-Engineering-Institute. *Software Architecture Glossary*.

> Available: http://www.sei.cmu.edu/architecture/start/glossary/index.cfm

[80]   Shergill, G. S. and Z. Chen, "Web-based Shopping: Consumers Attitudes Towards Online Shopping in New Zealand," *Journal of Electronic Commerce Research,* vol. 6, pp. 79-94, 2005.

[81]   Shim, S., V. Pendyala, et al., "Business-to-Business E-Commerce Frameworks," *IEEE Computer Society* vol. 33, pp. 40-47, 2000.

[82]   Software-architectures-Site. *Quality Attributes Scenarios*

        Available:
        http://www.softwarearchitectures.com/go/Discipline/DesigningArchitecture/Qua
        lityAttributes/Catalog/tabid/83/Default.aspx

[83]   Sundaram, M. and S. Shim, "Infrastructure for B2B Exchanges with RosettaNet," in *Third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, 2001, pp. 110-119.

[84]   Tassabehji, R., *Applying E-Commerce in Business*: SAGE Publications Ltd 2003.

[85]   Timmers, P., *Electronic Commerce: Strategies and Models for Business-to Business*: John Wiley & Sons, 2000.

[86]   Trastour, D., C. Preist, et al., "Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches," in *Proceedings of the 7th International Conference on Enterprise Distributed Object Computing (EDOC)*, 2003, pp. 222-231.

[87]   Turban, E., D. King, et al., *Electronic Commerce 2004: a Managerial Perspective*: Prentice Hall, 2004.

[88]   Turban, E., D. King, et al., *Electronic Commerce 2006: a Managerial Perspective*: Pearson Prentice Hall, 2006.

[89]   W3C. *Extensible Markup Language (XML)*.

        Available: http://www.w3.org/XML/

[90]   W3C, "XML Encryption Syntax and Processing," ed, 10 December 2002.

[91]   Wang, G. and S. Miller, "Intelligent Aggregation of Purchase Orders in E-Procurement," in *Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference* Enschede, The Netherlands, 2005, pp. 27-38.

[92]   Weerawarana, S. and F. Curbera. (2002). *Business Process with BPEL4WS: Understanding BPEL4WS, Part 1*.

        Available:

        http://www.ibm.com/developerworks/webservices/library/ws-bpelcol1/

[93]   Xiong-yi, L., "Research and Application of SOA in B2B Electronic Commerce," in *Proceedings of the 2009 International Conference on Computer Technology and Development*, USA 2009, pp. 649-653.

Appendices

Appendix A: Case study Working Environment

The Case study has been implemented using Java Web Services, the Requirement and software used for implementation of the Case Study is shows in Table A.1.

Software Requirement	Software Environment
Java Development Kit	JDK-6u18-windows-i586
.Net Development Environment	Microsoft Visual Studio 2008
Development Environment architecture	NetBeans IDE 6.7.1
Application Server	GlassFish Application Server 2.2
ESB	OpenESB/JBI jbi_components_istaller.jar
Engines and Binding Components	Open_esb_v2 jbi_components_istaller.jar BPEL-Engine ESB Composite Application
Data Base Connectivity	JDBC/ojdc6.jar
Data Access Layer	SQL-Server 2008

Table A.1: software used in implementation of the case study

Figure A.1depicts a snapshot of development environment with NetBeans IDE 6.7.1.
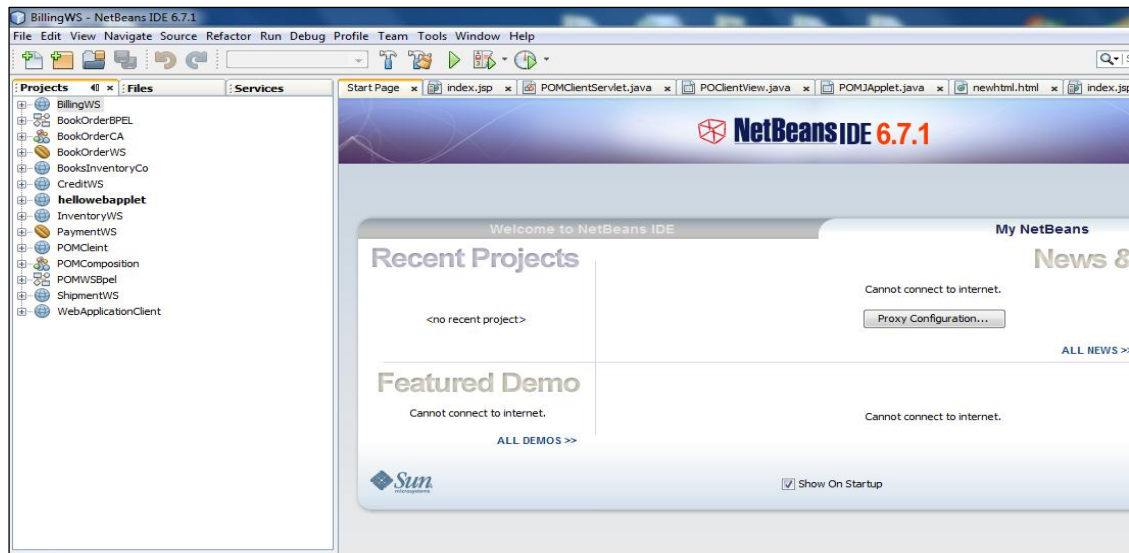
Figure A.1: Java Development Environment

Appendices

Figure A.2 depicts as snapshot of .Net development environment with Microsoft Visual Studio 2008

Figure A.2: .Net Development Environment

Appendix B: XML Schema Definition Type XSD of Purchase Order &Invoice

Figure A.3 depicts the XSD file for the Purchase Order Management required for the core services of POM, the schema definition contain basic customer, order and product type information. methods of delivery and method payment, which includes fields for credit card number, expiration date, and payment amount.

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

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/POSchema"
    xmlns:tns="http://xml.netbeans.org/schema/POSchema"
    elementFormDefault="qualified">
<xsd:complexType name="PurchaseOrder">
    <xsd:sequence>
        <xsd:element name="PersonalInformation" type="tns:PersonalInformation"></xsd:element>
        <xsd:element name="OrderInformation" type="tns:OrderInformation"></xsd:element>
        <xsd:element name="paymentInformation" type="tns:PaymentInformation"></xsd:element>
        <xsd:element name="ShippingInformation" type="tns:ShippingInformation"></xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="PersonalInformation">
    <xsd:sequence>
        <xsd:element name="Customer" type="xsd:string"></xsd:element>
        <xsd:element name="Address" type="xsd:string"></xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="OrderInformation">
    <xsd:sequence>
        <xsd:element name="Item" type="xsd:string"></xsd:element>
        <xsd:element name="Qty" type="xsd:int"></xsd:element>
        <xsd:element name="Price" type="xsd:int"></xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="PaymentInformation">
    <xsd:sequence>
        <xsd:element name="CCN" type="xsd:int"></xsd:element>
        <xsd:element name="ExpDate" type="xsd:string"></xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ShippingInformation">
    <xsd:sequence>
        <xsd:element name="Agent" type="xsd:string"></xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BillingReport">
    <xsd:sequence>
        <xsd:element name="Customer" type="xsd:string"></xsd:element>
        <xsd:element name="Item" type="xsd:string"></xsd:element>
        <xsd:element name="Price" type="xsd:int"></xsd:element>
        <xsd:element name="Qty" type="xsd:int"></xsd:element>
        <xsd:element name="ShippingCost" type="xsd:int"></xsd:element>
        <xsd:element name="OverallAmount" type="xsd:int"></xsd:element>
        <xsd:element name="ErrorReport" type="xsd:string"></xsd:element>
    </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```
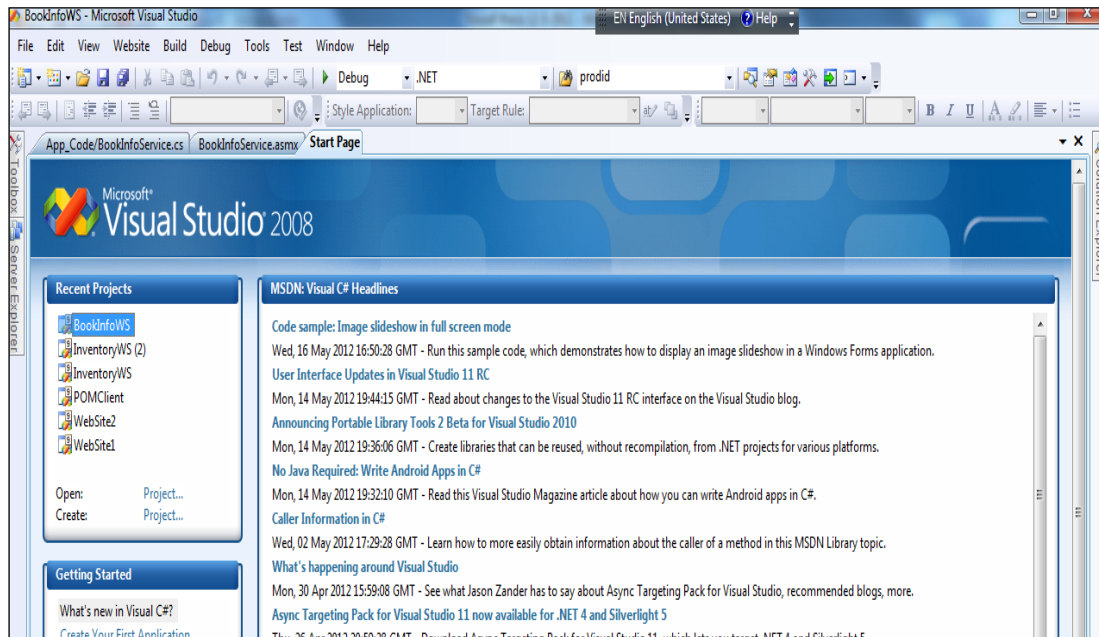
Figure A.3: Purchase Order XSD

The Figure A.4 depicts Snapshot of Purchase Order XSD Design

Figure A.4: Snapshot of Purchase Order XSD Design

Appendices

Figure A.5 depicts the XSD file for the Invoice, the invoice (Bill) schema defining how an invoice sent back to the client should be. The invoice schema definition contain purchase order date, billing information, order details (item list).
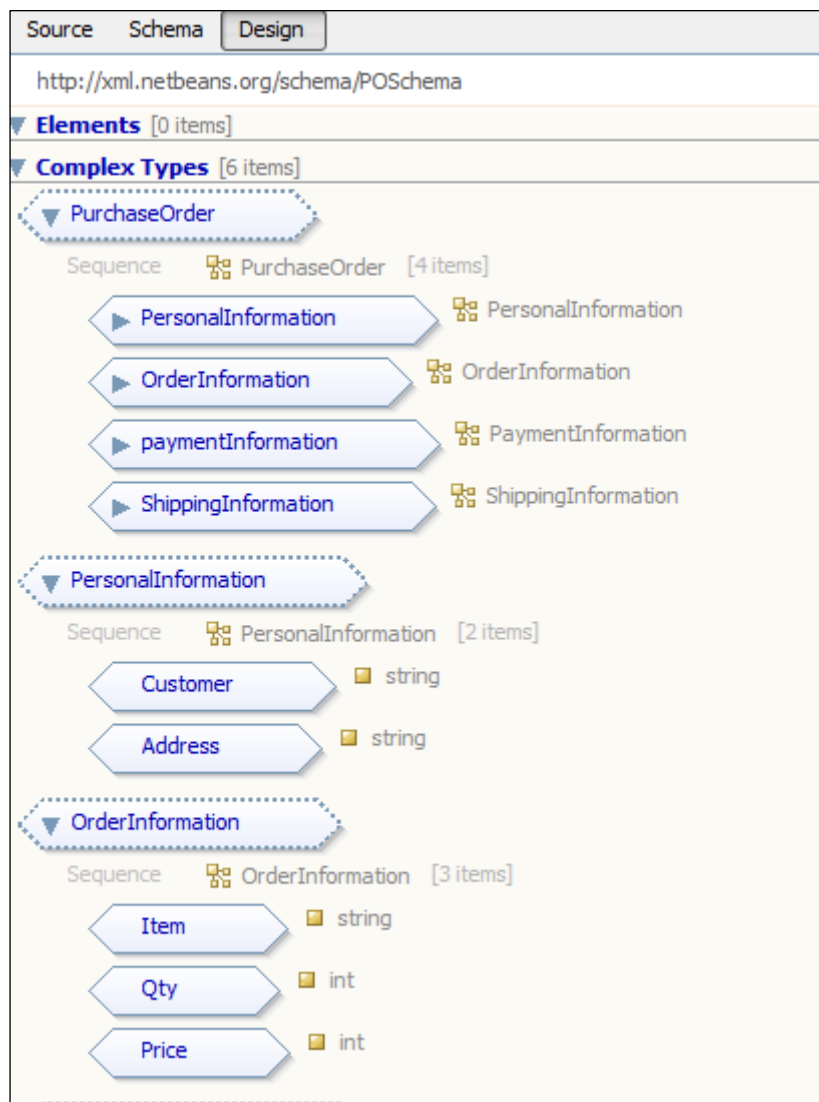
```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/examples/Invoice"
    xmlns:tns="http://xml.netbeans.org/examples/Invoice"
    elementFormDefault="qualified" xmlns:ns1="http://xml.netbeans.org/examples/PurchaseOrder">
    <xsd:import schemaLocation="PurchaseOrder.xsd" namespace="http://xml.netbeans.org/examples/PurchaseOrder"/>
    <xsd:element name="Invoice" type="tns:InvoiceType"/>

    <xsd:complexType name="InvoiceType">
        <xsd:sequence>
            <xsd:element name="items"  type="ns1:Items"/>
            <xsd:element name="orderDate" type="xsd:string"/>
            <xsd:element name="total" type="xsd:decimal"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Figure A.5 depicts the XSD file for the Invoice

Figure A.6 depicts Snapshot of Invoice XSD Design
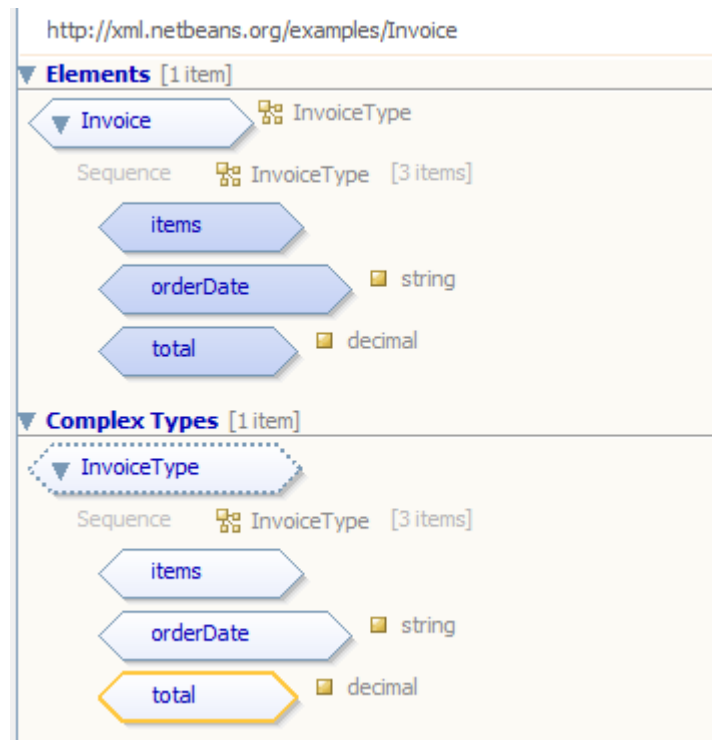
Figure A.6: Snapshot of Invoice XSD Design

Appendix C: Credit Check Web Service

▪ **WSDL file for the Credit Check Web service**

Figure A.7 shows the required WSDL File for the Credit Check Web Service, it has one operation ValidateCreditCard also has two messages ValidateCreditCard for Request and ValidateCreditCard Response.
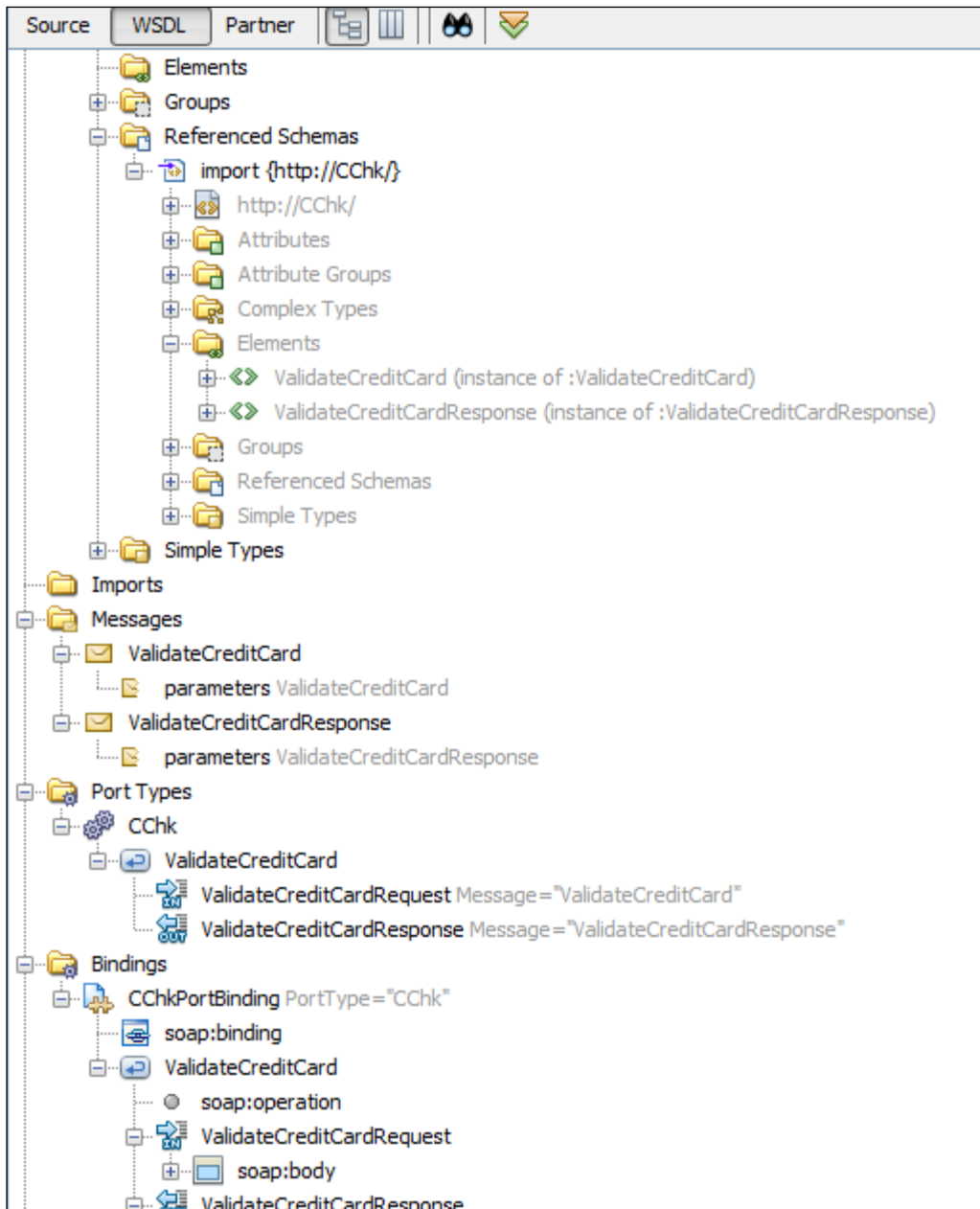
Figure A.7: WSDL File for Credit Check Web Service

- **The Credit Check Web Service Implementation**

The Following Figure A.8 depicts the Java Code for Credit Check Web Service

```
package CChk;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.text.DateFormat;
import java.text.ParseException;
/**
 *
 * @author yousef
 */
@WebService()
public class CChk {
 int CreditCards[]={1111,2222};
    String CCExpDates[]={"05-11-2012","10-01-2012"};
    int Credits[]={300,400};
    @WebMethod(operationName = "ValidateCreditCard")
    public String ValidateCreditCard(@WebParam(name = "CCN")
    int CCN,@WebParam(name = "ExpDate")
    String ExpDate , @WebParam(name = "Amount")
    int Amount) {
        //TODO write your implementation code here:
        for(int a=0;a<CreditCards.length;a++)
        {
            Date date=new Date();
            if(CreditCards[a]==CCN)
            {
                if(CCExpDates[a].compareTo(ExpDate)==0)
                {
                    try {
                        String str_date=ExpDate;
                        DateFormat formatter;

                        formatter = new SimpleDateFormat("dd-MM-yyyy");
                        date = (Date)formatter.parse(str_date);
                    }
                    catch(ParseException e)
                    {
                        return "Invalid";
                    }
                    if((date.getTime()-System.currentTimeMillis())>0)
                    {
                        if(Amount<=Credits[a])
                        return "ok";
                    }
                }
            }
        }
        return "Invalid";
    }
}
```

Figure A.8: Java Code for Credit Check Web Service

Appendix D: Inventory Check Web Service

▪ <u>**WSDL file for the Inventory Check Web service**</u>

Figure A.9 shows the required WSDL File for the Inventory Web Service, it has one operation INVChk, also has two messages the Check Item request and CheckItem Response.
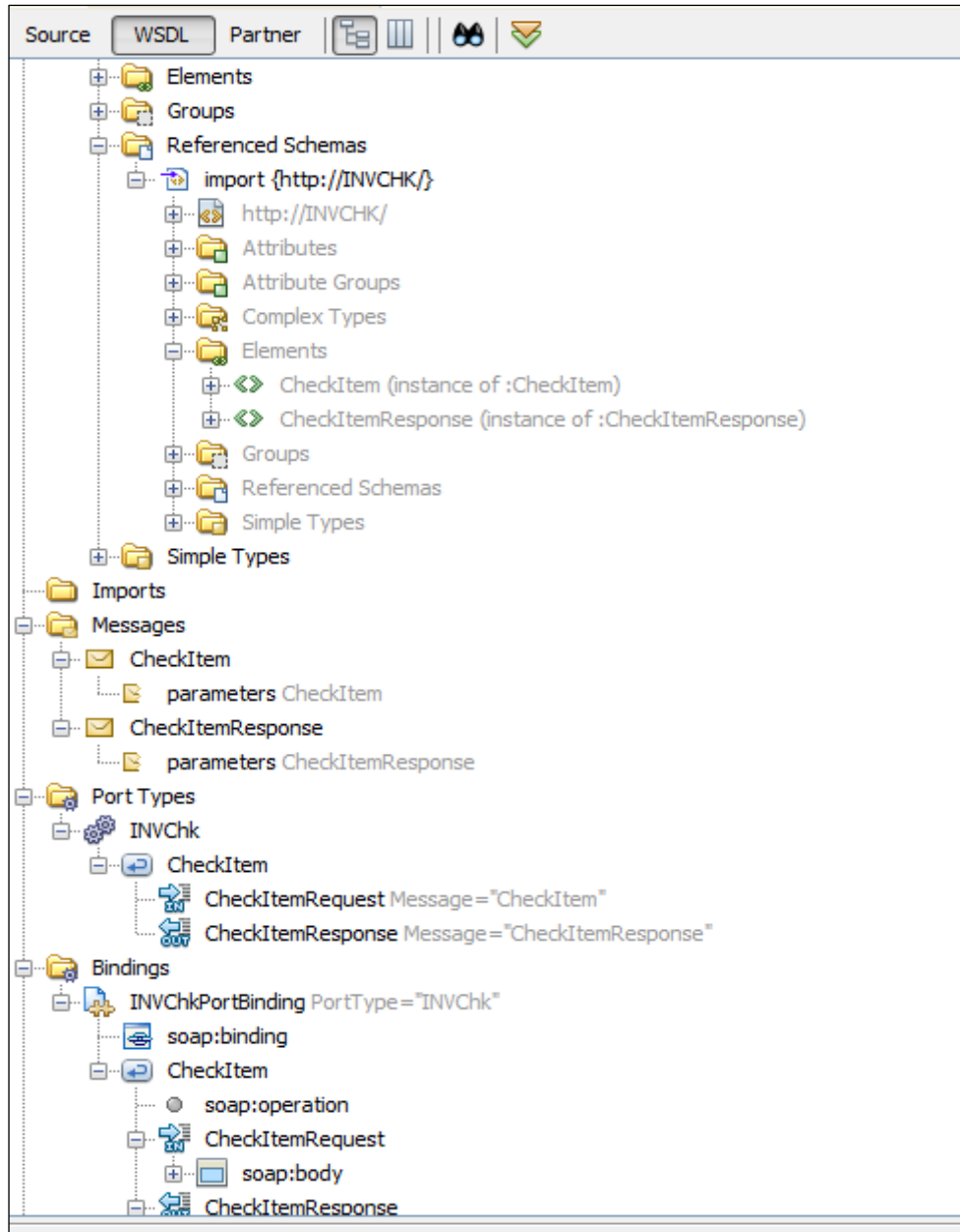
Figure A.9: WSDL File for Inventory Check Web Service

- **The Inventory Check Web Service Implementation**

Figure A.10 depicts the Java Code for Inventory Check Web Service

```
package INVCHK;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
// @author yousef
@WebService()
public class INVChk {
    String Items[]={"Java How to Program, 9/e","Web Services: Principles and Technology ",
                    "Software Engineering","A Grammar of the English Tongue "};
    int Quantities[]={10,30,14,35};
    /**
     * web service operation
     */
    @WebMethod(operationName = "CheckItem")
    public String CheckItem(@WebParam(name = "Item")
    String Item, @WebParam(name = "Qty")
    int Qty) {
        //TODO write your implementation code here:
        for(int a=0;a<Items.length;a++)
        {
            if(Items[a].compareTo(Item)==0)
                if(Qty<=Quantities[a])
                    return "ok";
        }
        return "failed";
    }
}
```

Table A.10: Java Code for Inventory check Web Service

Appendix E: Shipment Web Service

▪ <u>**WSDL file for the Shipment Web service**</u>

Figure A.11 shows the required WSDL File for the Shipment Web Service, it has one operation RequireShipping, also has two messages the RequireShippingRequest and RequireShippingResponse.
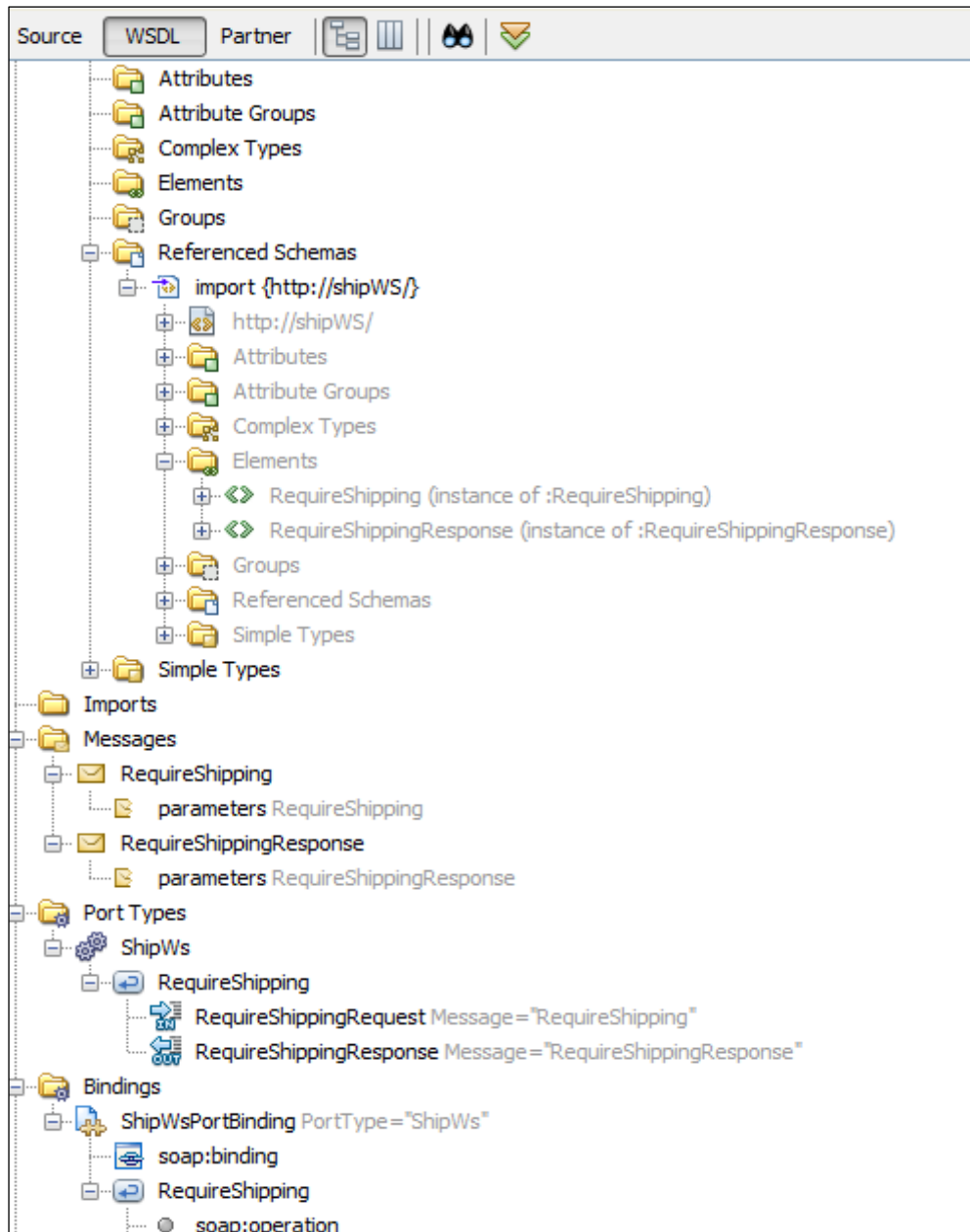
Figure A.11: WSDL File for Shipment Web Service

▪ **The Shipment Web Service Implementation**

Figure A.12 depicts the Java Code for Shipment Web Service

```java
package shipWS;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
/**
 *
 * @author yousef
 */
@WebService()
public class ShipWS {
String Agents[]={"Yousef","IUG"};
    String TargetAddresses[]={"Palestine -- Gaza","Palestine -- Rafah"};
    int Costs[]={10,30};

    /**
     * Web service operation
     */
    @WebMethod(operationName = "RequireShipping")
    public int RequireShipping(@WebParam(name = "Agent")
    String Agent, @WebParam(name = "Address")
    String Address) {
        //TODO write your implementation code here:

        for(int a=0;a<Agents.length;a++)
        {
            if(Agents[a].compareTo(Agent)==0)

                // if(TargetAddresses[a].compareTo(Address)==0)
                        return Costs[a];
        }
        return -1;

    }
}
```

Figure A.12: Java Code for Shipment Web Service

Appendix F: Billing Web Service

- ### WSDL file for the Billing Web service

Figure A.13 shows the required WSDL File for the Billing Web Service, it has one operation BillWS, also has two messages GenerateInviceRequest, GenerateInviceResponse.
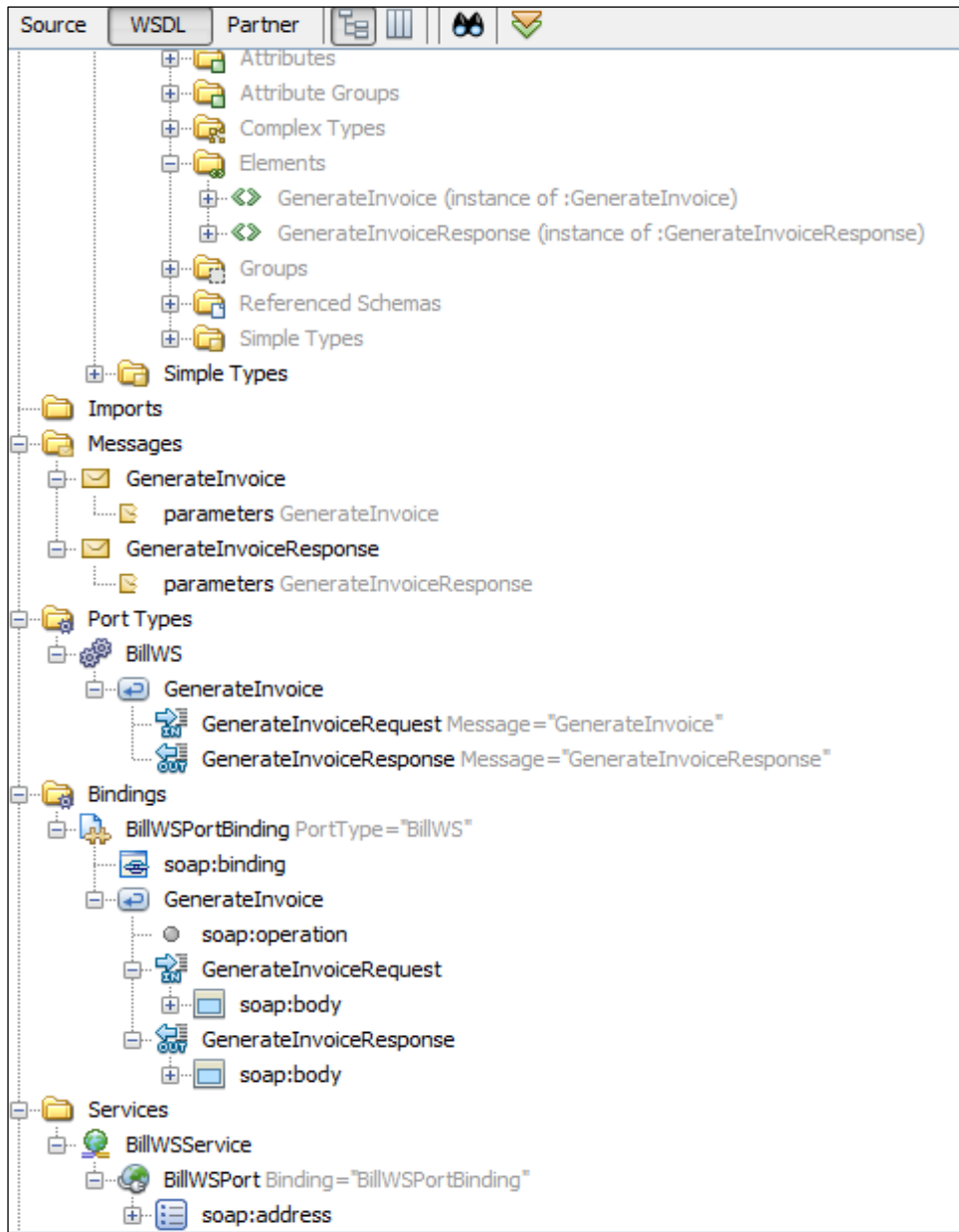
Figure A.13: WSDL File for Billing Web Service

- **The Billing Web Service Implementation**

Figure A.14 depicts the Java Code for Billing Web Service.

```
package BillWS;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
/**
 *
 * @author yousef
 */
@WebService()
public class BillWS {
@WebMethod(operationName = "GenerateInvoice")
    public int GenerateInvoice(@WebParam(name = "Qty")
    int Qty, @WebParam(name = "Price")
    int Price, @WebParam(name = "ShippingCost")
    int ShippingCost) {
        //TODO write your implementation code here:
        return Price*Qty+ShippingCost;
    }
}
```

Table A.14: Java Code for Billing Web Service

Appendix G: The Composite Application for POM Service and BPEL

▪ <u>**WSDL file for Composite Service for POM Service**</u>

Figure A.15 depicts the required WSDL File for the PO Web Service, it has one operation POOperation, also has two messages the POOperationRequest, and POOperationRespons. This WSDL File For composite service/process that orchestrates the core services of POM (Credit Check, Inventory Check, Billing, and Shipment) Web services.
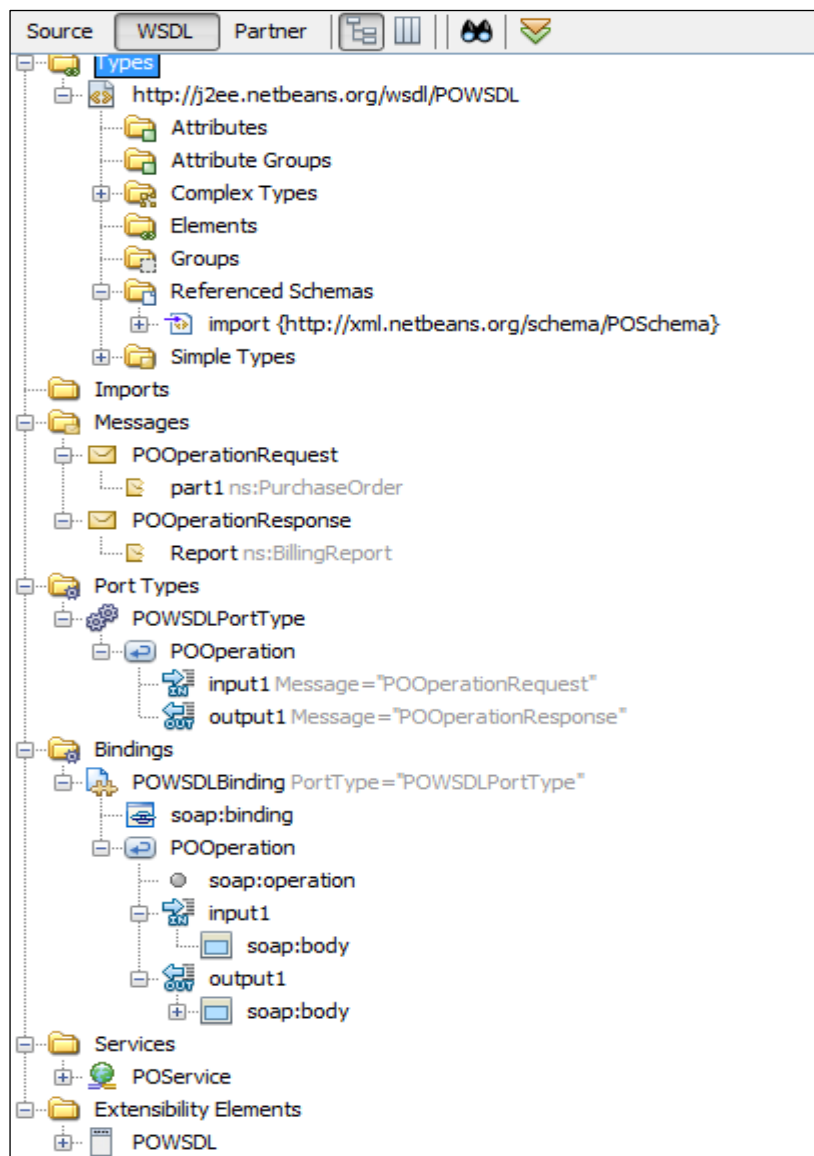
Figure A.15: WSDL File for POM Composite Service

▪ **BPEL Design View For POM Composite Service**

The following view Based on four services implemented above, It execute the process from request the item product and submission of order based on the XSD of Purchase Order, to return an Invoice based on the XSD Invoice. Figure A.16 depicts BPEL Design for POM Composite Web Service. In this view core services of POM (Credit Check, Inventory Check, Billing, and Shipment) are to right of Figure A.16, and the POM composite Web Service to the left. The BPEL source code for POM composite web services is depicts in Figures A.16 (Part3- to- Part6).
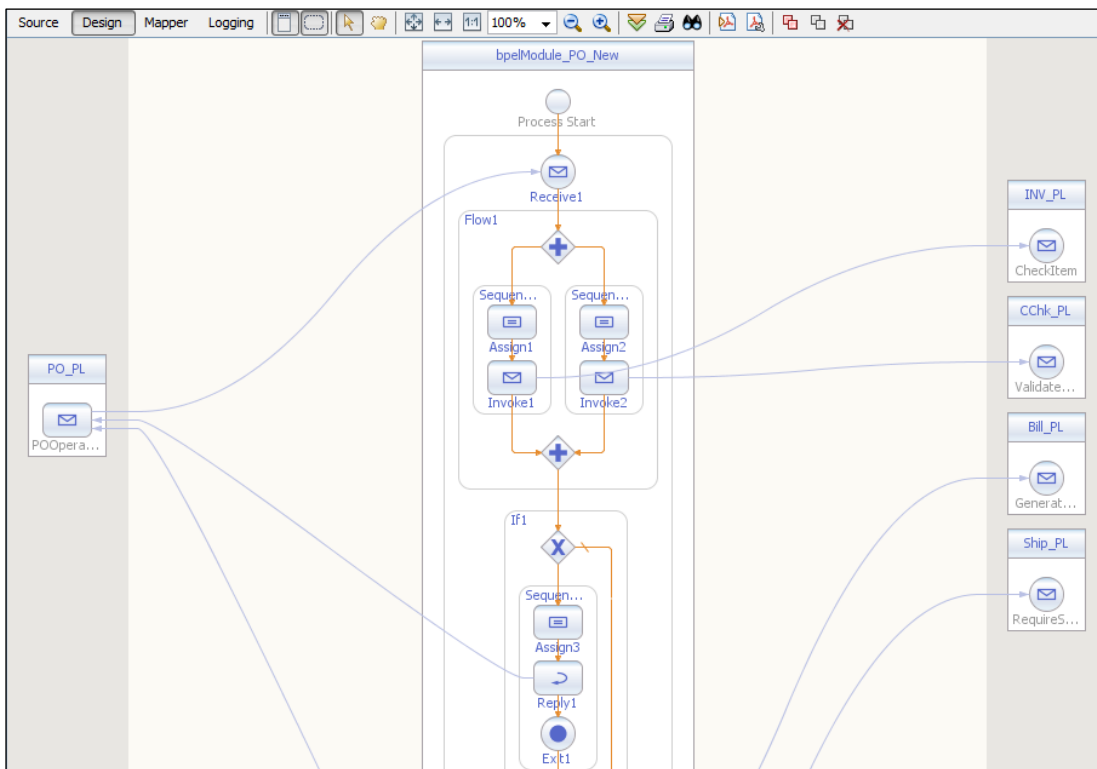
Figure A.16 (Part1): BPEL Design for POM Composite Web Service
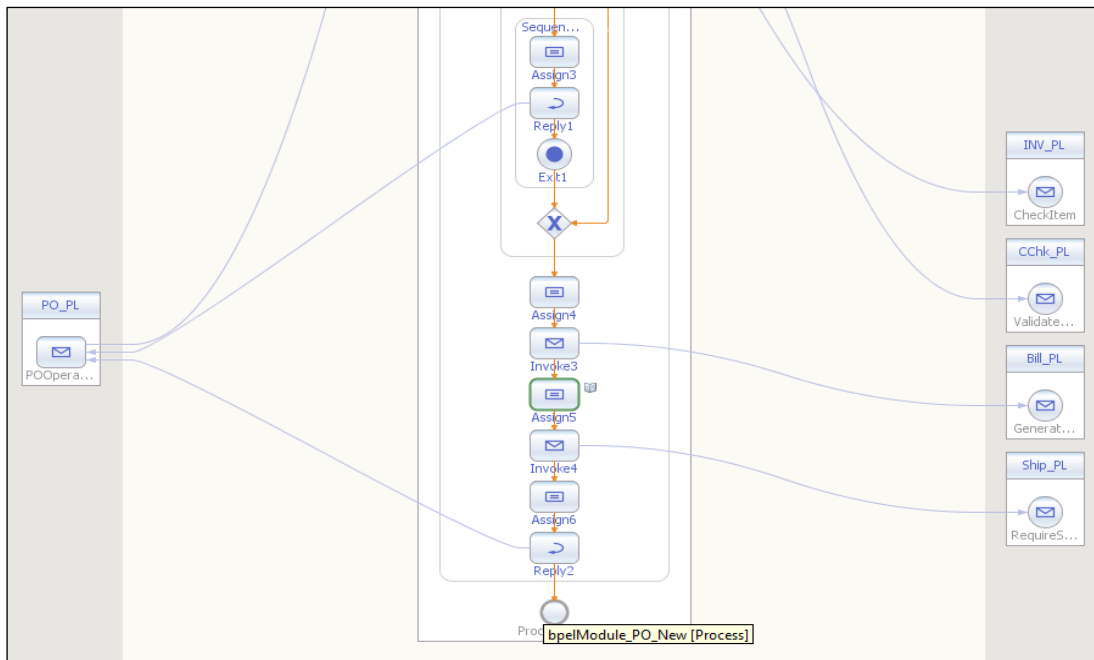
Figure APP.16 (Part2): BPEL Design for POM Composite Web Service

```xml
<?xml version="1.0" encoding="UTF-8"?>
<process
    name="bpelModule_PO_New"
    targetNamespace="http://enterprise.netbeans.org/bpel/BpelModule_PO_New/bpelModule_PO_New"
    xmlns:tns="http://enterprise.netbeans.org/bpel/BpelModule_PO_New/bpelModule_PO_New"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:sxt="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Trace"
    xmlns:sxed="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Editor"
    xmlns:sxeh="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/ErrorHandling"
xmlns:sxed2="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Editor2"
xmlns:ns0="http://xml.netbeans.org/schema/POSchema">
    <import namespace="http://j2ee.netbeans.org/wsdl/POWSDL" location="POWSDL.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://enterprise.netbeans.org/bpel/INVChkServiceWrapper"
location="INVChkServiceWrapper.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://INVCHK/" location="http://localhost:8080/INVChk/INVChkService?WSDL"
importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://enterprise.netbeans.org/bpel/CChkServiceWrapper" location="CChkServiceWrapper.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://CChk/" location="http://localhost:8080/CChk/CChkService?WSDL"
importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://enterprise.netbeans.org/bpel/BillWSServiceWrapper"
location="BillWSServiceWrapper.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://BillWS/" location="http://localhost:8080/BillWS/BillWSService?WSDL"
importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://enterprise.netbeans.org/bpel/ShipWsServiceWrapper"
```

Figure APP.16 (Part3): BPEL Source Code for POM Composite Web Service

```
location="ShipWsServiceWrapper.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://shipWS/" location="http://localhost:8080/shipWS/ShipWsService?WSDL"
importType="http://schemas.xmlsoap.org/wsdl/"/>
  <partnerLinks>
    <partnerLink name="INV_PL" xmlns:tns="http://enterprise.netbeans.org/bpel/INVChkServiceWrapper"
partnerLinkType="tns:INVChkLinkType" partnerRole="INVChkRole"/>
    <partnerLink name="CChk_PL" xmlns:tns="http://enterprise.netbeans.org/bpel/CChkServiceWrapper"
partnerLinkType="tns:CChkLinkType" partnerRole="CChkRole"/>
    <partnerLink name="Bill_PL" xmlns:tns="http://enterprise.netbeans.org/bpel/BillWSServiceWrapper"
partnerLinkType="tns:BillWSLinkType" partnerRole="BillWSRole"/>
    <partnerLink name="Ship_PL" xmlns:tns="http://enterprise.netbeans.org/bpel/ShipWsServiceWrapper"
partnerLinkType="tns:ShipWsLinkType" partnerRole="ShipWsRole"/>
    <partnerLink name="PO_PL" xmlns:tns="http://j2ee.netbeans.org/wsdl/POWSDL" partnerLinkType="tns:POWSDL"
myRole="POWSDLPortTypeRole"/>
  </partnerLinks>
  <variables>
    <variable name="POOperationOut2" xmlns:tns="http://j2ee.netbeans.org/wsdl/POWSDL"
messageType="tns:POOperationResponse"/>
    <variable name="RequireShippingOut" xmlns:tns="http://shipWS/" messageType="tns:RequireShippingResponse"/>
    <variable name="RequireShippingIn" xmlns:tns="http://shipWS/" messageType="tns:RequireShipping"/>
    <variable name="GenerateInvoiceOut" xmlns:tns="http://BillWS/" messageType="tns:GenerateInvoiceResponse"/>
    <variable name="GenerateInvoiceIn" xmlns:tns="http://BillWS/" messageType="tns:GenerateInvoice"/>
    <variable name="POOperationOut" xmlns:tns="http://j2ee.netbeans.org/wsdl/POWSDL"
messageType="tns:POOperationResponse"/>
    <variable name="ValidateCreditCardOut" xmlns:tns="http://CChk/"
messageType="tns:ValidateCreditCardResponse"/>
    <variable name="ValidateCreditCardIn" xmlns:tns="http://CChk/" messageType="tns:ValidateCreditCard"/>
    <variable name="CheckItemOut" xmlns:tns="http://INVCHK/" messageType="tns:CheckItemResponse"/>
    <variable name="CheckItemIn" xmlns:tns="http://INVCHK/" messageType="tns:CheckItem"/>
    <variable name="POOperationIn" xmlns:tns="http://j2ee.netbeans.org/wsdl/POWSDL"
messageType="tns:POOperationRequest"/>
  </variables>
  <sequence>
    <receive name="Receive1" createInstance="yes" partnerLink="PO_PL" operation="POOperation"
xmlns:tns="http://j2ee.netbeans.org/wsdl/POWSDL" portType="tns:POWSDLPortType" variable="POOperationIn"/>
    <flow name="Flow1">
      <sequence name="Sequence1">
        <assign name="Assign1">
          <copy>
            <from>$POOperationIn.part1/ns0:OrderInformation/ns0:Item</from>
            <to>$CheckItemIn.parameters/Item</to>
          </copy>
          <copy>
            <from>$POOperationIn.part1/ns0:OrderInformation/ns0:Qty</from>
            <to>$CheckItemIn.parameters/Qty</to>
          </copy>
        </assign>
        <invoke name="Invoke1" partnerLink="INV_PL" operation="CheckItem" xmlns:tns="http://INVCHK/"
portType="tns:INVChk" inputVariable="CheckItemIn" outputVariable="CheckItemOut"/>
```

Figure APP.16 (Part4): BPEL Source Code for POM Composite Web Service

```
      </sequence>
        <sequence name="Sequence2">
          <assign name="Assign2">
            <copy>
              <from>$POOperationIn.part1/ns0:paymentInformation/ns0:CCN</from>
              <to>$ValidateCreditCardIn.parameters/CCN</to>
            </copy>
            <copy>
              <from>$POOperationIn.part1/ns0:paymentInformation/ns0:ExpDate</from>
              <to>$ValidateCreditCardIn.parameters/ExpDate</to>
            </copy>
            <copy>
              <from>$POOperationIn.part1/ns0:OrderInformation/ns0:Qty *
$POOperationIn.part1/ns0:OrderInformation/ns0:Price</from>
                <to>$ValidateCreditCardIn.parameters/Amount</to>
            </copy>
          </assign>
          <invoke name="Invoke2" partnerLink="CChk_PL" operation="ValidateCreditCard" xmlns:tns="http://CChk/"
portType="tns:CChk" inputVariable="ValidateCreditCardIn" outputVariable="ValidateCreditCardOut"/>
        </sequence>
      </flow>
      <if name="If1">
        <condition>$CheckItemOut.parameters/return = 'failed' or $ValidateCreditCardOut.parameters/return =
'Invalid'</condition>
          <sequence name="Sequence3">
            <assign name="Assign3">
              <copy>
                <from>'Inventory Empty or Credit Card Not Valid'</from>
                <to>$POOperationOut.Report/ns0:ErrorReport</to>
              </copy>
            </assign>
            <reply name="Reply1" partnerLink="PO_PL" operation="POOperation"
xmlns:tns="http://j2ee.netbeans.org/wsdl/POWSDL" portType="tns:POWSDLPortType" variable="POOperationOut"/>
            <exit name="Exit1"/>
          </sequence>
      </if>
      <assign name="Assign4">
        <copy>
          <from>$POOperationIn.part1/ns0:OrderInformation/ns0:Qty</from>
          <to>$GenerateInvoiceIn.parameters/Qty</to>
        </copy>
        <copy>
          <from>$POOperationIn.part1/ns0:OrderInformation/ns0:Price</from>
          <to>$GenerateInvoiceIn.parameters/Price</to>
        </copy>
        <copy>
          <from>0</from>
          <to>$GenerateInvoiceIn.parameters/ShippingCost</to>
        </copy>
```

Figure APP.16 (Part5): BPEL Source Code for POM Composite Web Service

```
</assign>
    <invoke name="Invoke3" partnerLink="Bill_PL" operation="GenerateInvoice" xmlns:tns="http://BillWS/"
portType="tns:BillWS" inputVariable="GenerateInvoiceIn" outputVariable="GenerateInvoiceOut"/>
    <assign name="Assign5">
      <copy>
        <from>$POOperationIn.part1/ns0:PersonalInformation/ns0:Address</from>
        <to>$RequireShippingIn.parameters/Address</to>
      </copy>
      <copy>
        <from>$POOperationIn.part1/ns0:ShippingInformation/ns0:Agent</from>
        <to>$RequireShippingIn.parameters/Agent</to>
      </copy>
    </assign>
    <invoke name="Invoke4" partnerLink="Ship_PL" operation="RequireShipping" xmlns:tns="http://shipWS/"
portType="tns:ShipWs" inputVariable="RequireShippingIn" outputVariable="RequireShippingOut"/>
    <assign name="Assign6">
      <copy>
        <from>$POOperationIn.part1/ns0:OrderInformation/ns0:Price</from>
        <to>$POOperationOut2.Report/ns0:Price</to>
      </copy>
      <copy>
        <from>$POOperationIn.part1/ns0:OrderInformation/ns0:Item</from>
        <to>$POOperationOut2.Report/ns0:Item</to>
      </copy>
      <copy>
        <from>$POOperationIn.part1/ns0:OrderInformation/ns0:Qty</from>
        <to>$POOperationOut2.Report/ns0:Qty</to>
      </copy>
      <copy>
        <from>$POOperationIn.part1/ns0:PersonalInformation/ns0:Customer</from>
        <to>$POOperationOut2.Report/ns0:Customer</to>
      </copy>
      <copy>
        <from>$RequireShippingOut.parameters/return</from>
        <to>$POOperationOut2.Report/ns0:ShippingCost</to>
      </copy>
      <copy>
        <from>$RequireShippingOut.parameters/return + $GenerateInvoiceOut.parameters/return</from>
        <to>$POOperationOut2.Report/ns0:OverallAmount</to>
      </copy>
      <copy>
        <from>'NoErrors'</from>
        <to>$POOperationOut2.Report/ns0:ErrorReport</to>
      </copy>
    </assign>
    <reply name="Reply2" partnerLink="PO_PL" operation="POOperation"
xmlns:tns="http://j2ee.netbeans.org/wsdl/POWSDL" portType="tns:POWSDLPortType" variable="POOperationOut2"/>
  </sequence>
</process>
```

Figure APP.16 (Part6): BPEL Source Code for POM Composite Web Service

- **Service Assembly of POM Composite Web Service**

The service Assembly of POM Composition is depicts in Figure A.17
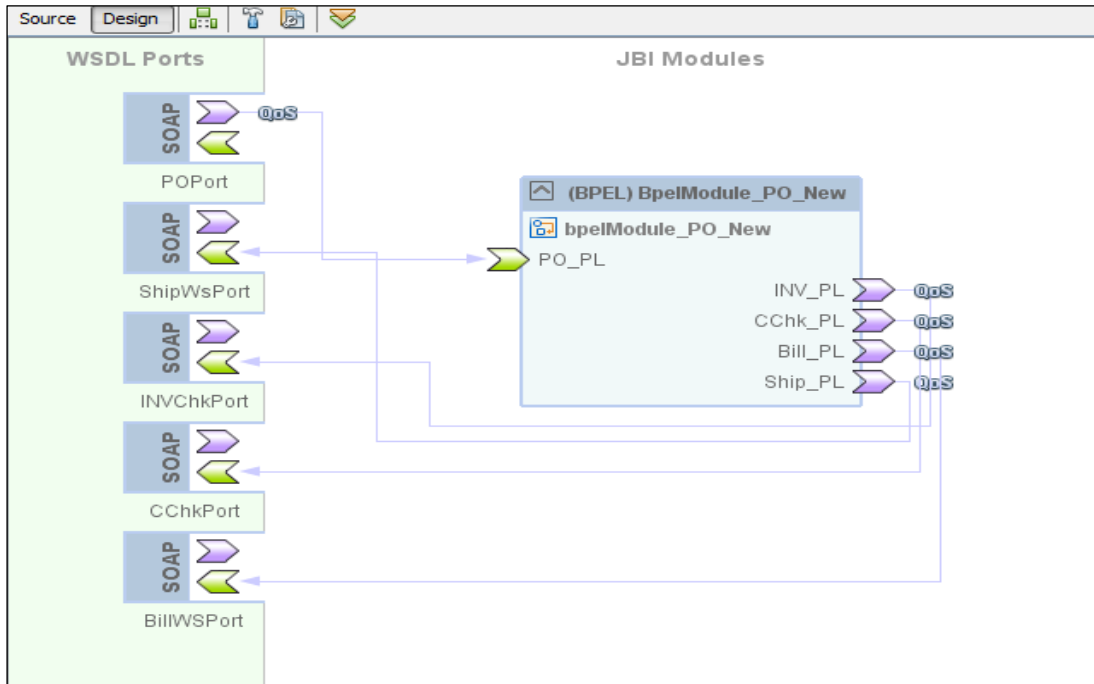
Figure A.17: Service Assembly of POM Composition

Appendix H: BookInfo Web Service .NET with C#

In this appendix we provide all related code that we construct the service of Book Information with Database created with SQL-Server 2008

- ▪ **DAL (Data Access Layer) to access database**

Data Access Layer is set of classes used to access database. We centralize the entire data access to DAL. Figure A.18 show book information table which is created in *msbookdb* database of SQL Server Express Edition.
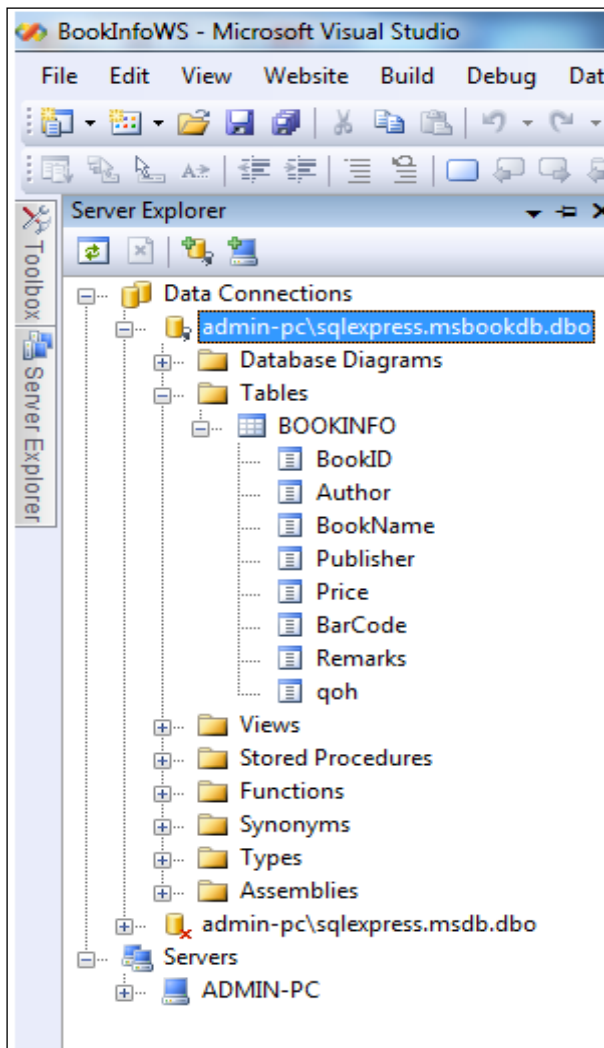
Figure A.18: SQL Server Book Information database

▪ **BookInfoService.cs**

Figure A.19 .Net code Web service for book information service which makes use of
Data Access Layer (DAL).

```csharp
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Linq;


[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]


public class BookInfoService : System.Web.Services.WebService {

    [WebMethod(Description = "Returns Details Of All BOOKs")]
    public BookInfo GetAllBookInfo()
    {
        return BookInfoDAL.GetAllBookInfo();
    }

    [WebMethod(Description = "Returns Details Of A Single Book ")]
    public BookInfo GetBookInfo(int BookID)
    {
        return BookInfoDAL.GetBookInfo(BookID);
    }

}
```

Figure A.19: .Net Book Information Web Service

▪ **BookInfoDAL.cs**

Figure A.20 which contains static methods for operations related to book information
database that access this database through DAL.

```
sing System.Web.UI;
sing System.Web.UI.HtmlControls;
sing System.Web.UI.WebControls;
sing System.Web.UI.WebControls.WebParts;
sing System.Xml.Linq;
sing System.Data.SqlClient;

// Summary description for BookInfoDAL
ublic class BookInfoDAL

        public static BookInfo GetAllBookInfo()

   {
        SqlConnection con = new SqlConnection(Database.ConnectionString);
        con.Open();
        SqlCommand cmd = new SqlCommand("select * from BOOKINFO", con);
        SqlDataReader dr = cmd.ExecuteReader();
        BookInfo Bookinfo = new BookInfo();
        //Product products = new Product();
        while (dr.Read())
        {
            BookInfo P = new BookInfo();
            P.BookID = (int)dr["BookID"];
            P.Author = dr["Author"].ToString();
            P.Bookname = dr["BookName"].ToString();
            P.Publisher = dr["Publisher"].ToString();
            P.BarCode = dr["BarCode"].ToString();
            P.Remarks = dr["remarks"].ToString();
            P.Qty = (int)dr["qoh"];
            P.Price = Convert.ToDouble(dr["price"]);

        }
        dr.Close();
        con.Close();
        return Bookinfo;
   }
   public static BookInfo GetBookInfo(int BookID)
   {
        SqlConnection con = new SqlConnection(Database.ConnectionString);
        con.Open();
        SqlCommand cmd = new SqlCommand("select * from BOOKINFO where BookID = @BookID", con);
        cmd.Parameters.AddWithValue("@BookID", BookID);
        SqlDataReader dr = cmd.ExecuteReader();
        if (dr.Read())
        {
            BookInfo P = new BookInfo();
            P.BookID    = (int)dr["BookID"];
            P.Author    = dr["Author"].ToString();
            P.Bookname  = dr["BookName"].ToString();
            P.Publisher = dr["Publisher"].ToString();
            P.BarCode   = dr["BarCode"].ToString();
            P.Remarks = dr["remarks"].ToString();
            P.Qty = (int)dr["qoh"];
            P.Price = Convert.ToDouble(dr["price"]);
            return P;
        }

        else  // product not found
            return null;
   }
```

Figure A.20: .Net Method Related to Book Information Database

- **BookInfo.cs**

Figure A.21 depicts .Net code to represent data for book information

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

/// Summary description for BookInfo
public class BookInfo
{
    private int bookID, qty;
    private string author,publisher,bookname,barCode, remarks;
    private double price;
    public int Qty
    {
        get { return qty; }
        set { qty = value; }
    }
    public int BookID
    {
        get { return bookID; }
        set { bookID = value; }
    }
    public string Author
    {
        get { return author; }
        set { author = value; }
    }
    public string Publisher
    {
        get { return publisher; }
        set { publisher = value; }
    }
    public string Bookname
    {
        get { return bookname; }
        set { bookname = value; }
    }
    public string BarCode
    {
        get { return barCode; }
        set { barCode = value; }
    }
    public string Remarks
    {
        get { return remarks; }
        set { remarks = value; }
    }
    public double Price
    {
        get { return price; }
        set { price = value; }
    }
}
```

Figure A.21: .Net Code to Represent Data for Book Information

- **Database.cs class**

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Web.Configuration;

/// <summary>
/// Summary description for Database
/// </summary>
public class Database
{
    public static string ConnectionString
    {
        get
        {
            return WebConfigurationManager.ConnectionStrings["msdbcs"].ConnectionString;
        }
    }
}
```

Figure A.22: C# Code for Connection Book Info Table

- **Configuring the connection to the database in Web.config**

```
<connectionStrings>
    <add name="msdbcs" connectionString="Data Source=.\SQLEXPRESS;Initial Catalog=msbookdb;Integrated Security=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Figure A.22: Connection to Database for Book Information Service

- **WSDL FILE**

Figure A.23 depicts WSDL document for the book information Web service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:tns="http://www.BookstorShop.com/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetNamespace="http://www.BookstorShop.com/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.BookstorShop.com/">
      <s:element name="GetAllBookInfo">
        <s:complexType />
      </s:element>
      <s:element name="GetAllBookInfoResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetAllBookInfoResult" type="tns:BookInfo" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="BookInfo">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="Qty" type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="BookID" type="s:int" />
          <s:element minOccurs="0" maxOccurs="1" name="Author" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Publisher" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Bookname" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="BarCode" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Remarks" type="s:string" />
          <s:element minOccurs="1" maxOccurs="1" name="Price" type="s:double" />
        </s:sequence>
      </s:complexType>
      <s:element name="GetBookInfo">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="BookID" type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetBookInfoResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetBookInfoResult" type="tns:BookInfo" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="GetAllBookInfoSoapIn">
    <wsdl:part name="parameters" element="tns:GetAllBookInfo" />
  </wsdl:message>
  <wsdl:message name="GetAllBookInfoSoapOut">
    <wsdl:part name="parameters" element="tns:GetAllBookInfoResponse" />
  </wsdl:message>
  <wsdl:message name="GetBookInfoSoapIn">
    <wsdl:part name="parameters" element="tns:GetBookInfo" />
  </wsdl:message>
  <wsdl:message name="GetBookInfoSoapOut">
    <wsdl:part name="parameters" element="tns:GetBookInfoResponse" />
  </wsdl:message>
```

Figure A.23: (Part1) WSDL File for Book Information Service

```
<wsdl:portType name="BookInfoServiceSoap">
  <wsdl:operation name="GetAllBookInfo">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Returns Details Of All BOOKs</wsdl:documentation>
    <wsdl:input message="tns:GetAllBookInfoSoapIn" />
    <wsdl:output message="tns:GetAllBookInfoSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="GetBookInfo">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Returns Details Of A Single Book </wsdl:documentation>
    <wsdl:input message="tns:GetBookInfoSoapIn" />
    <wsdl:output message="tns:GetBookInfoSoapOut" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="BookInfoServiceSoap" type="tns:BookInfoServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="GetAllBookInfo">
    <soap:operation soapAction="http://www.BookstorShop.com/GetAllBookInfo" style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetBookInfo">
    <soap:operation soapAction="http://www.BookstorShop.com/GetBookInfo" style="cocument" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="BookInfoServiceSoap12" type="trs:BookInfoServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="GetAllBookInfo">
    <soap12:operation soapAction="http://www.BookstorShop.com/GetAllBookInfo" style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetBookInfo">
    <soap12:operation soapAction="http://www.BookstorShop.com/GetBookInfo" style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="BookInfoService">
  <wsdl:port name="BookInfoServiceSoap" binding="trs:BookInfoServiceSoap">
    <soap:address location="http://localhost:54778/BookInfoWS/BookInfoService.asmx" />
  </wsdl:port>
  <wsdl:port name="BookInfoServiceSoap12" binding="tns:BookInfoServiceSoap12">
    <soap12:address location="http://localhost:54778/BookInfoWS/BcokInfoService.asmx" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Figure A.23: (Part2) WSDL File for Book Information Service

Appendix I: The Bookstore Interface as Front End Access interface

The Front End Access interface is written in JSP and runs in the context of the Web application service GlassFish. The interface allows customer to view book information that return all related information about the book in the BookInfoService which is .Net Web service is consumed with the environment Java NetBeanes Figure A.24 is Snapshot of consuming .Net service (BookInfoService) and POM Composite service which contain four services (Credit Checking, Inventory Checking, Shipment, and billing).
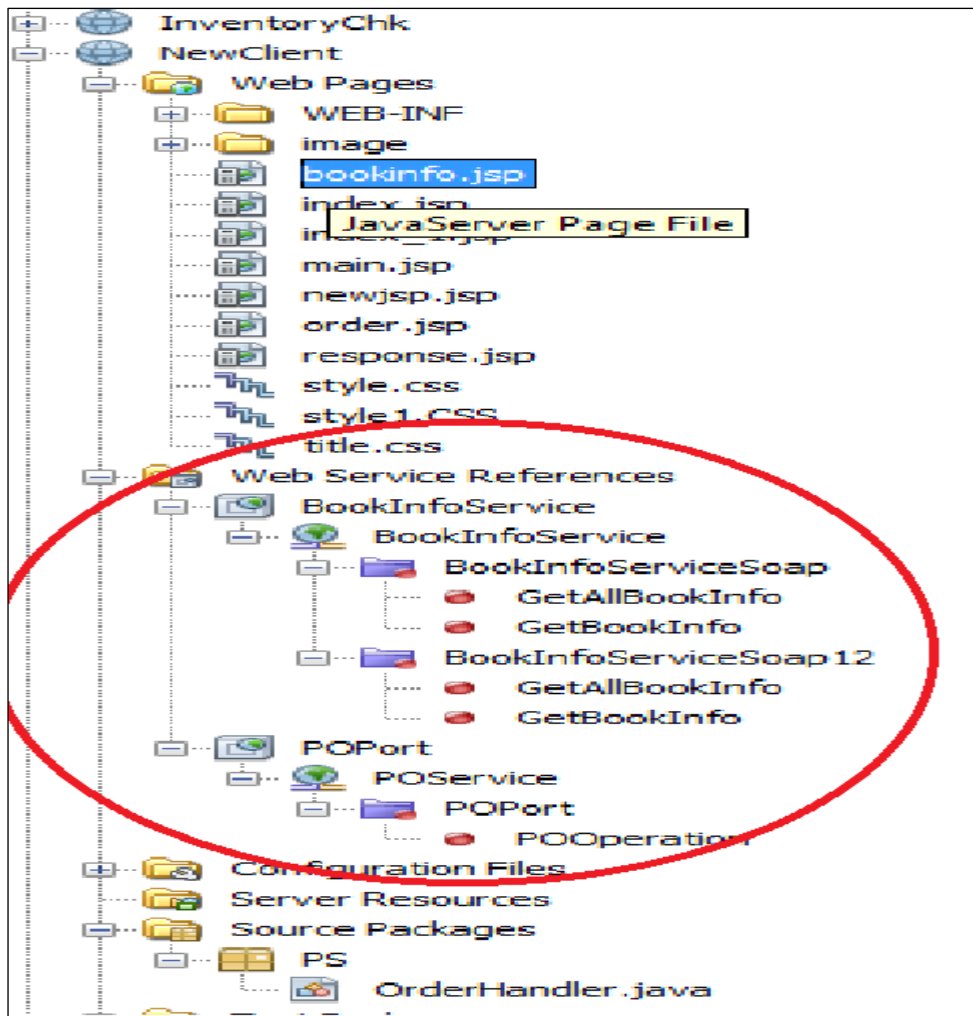
Figure A.24: Client Service to Consume Java Services and .Net Service

The Core Interface of Case study is writing in JSP language that consume the .Net service Figure A.25 is present snapshots of Book Store Shop

Figure A.25: Front End Interface of Book Store Shop

Book Info accessing .Net Service

In Figure A.26 present and give result of accessing .Net Service that returned when customer click on the book info in the link of page of Front End interface which invokes and consume the .Net Service will return JSP Page contain all information about the book ware selected.
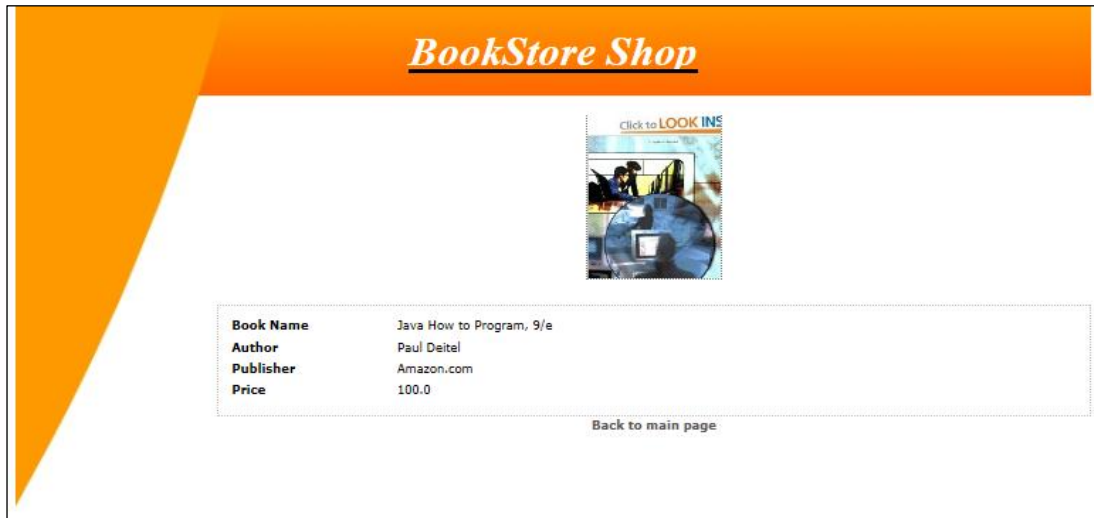
Figure A.26: Front End Interface Result of Accessing .Net Service Present the Information of Book Selected

Figure A.27 depicts the JSP Code for the Front End interface to access the BookInfo service.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Book Information</title>
        <link rel="stylesheet" href="style.css" type="text/css" charset="utf-8" />
        <LINK REL="STYLESHEET" HREF="style1.css" TYPE="TEXT/CSS">
        <style type="text/css"></style>
        <%
            com.bookstorshop.BookInfo result = null;
            int bookID = Integer.parseInt(request.getParameter("bookId"));
            try {
                com.bookstorshop.BookInfoService service = new com.bookstorshop.BookInfoService();
                com.bookstorshop.BookInfoServiceSoap port = service.getBookInfoServiceSoap12();
                result = port.getBookInfo(bookID);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        %>
    </head>
    <body bgcolor="ffffff">
        <center>
            <table border="0" cellpadding="0" cellspacing="0">
                <tr>
                    <td width="800" align="center" colspan="3" background="image/navtop.jpg" height="75">
                        <i><u><b><font size="+3" color="white" face="Aquaduct"> BookStore Shop </font></b></u></i></td>
                </tr>
            </table>
            <table border="0" cellpadding="0" cellspacing="0">
                <tr>
                    <td width="135" align="center" colspan="3" background="image/navleft.jpg"
                        style="background-repeat:no-repeat " height="328" valign="top"></td>
                    <td width="15"> </td>
                    <td width="650" valign="top" colspan="3" height="328"><br>
```

Figure A.27 (Part1): JSP Code access the Book Information service

```
<div class="bookartcontainer">
    <div class="bookart">
        <img src="image/books/<%= bookID%>.jpg" />
    </div>
</div>
<div class="bookinfo">

    <span class="infopiece">
        <span class="label">Book Name</span>
        <span class="value"><%= result.getBookname()%></span>
    </span>
    <span class="infopiece">
        <span class="label">Author</span>
        <span class="value"><%= result.getAuthor()%></span>
    </span>
    <span class="infopiece">
        <span class="label">Publisher</span>
        <span class="value"><%= result.getPublisher()%></span>
    </span>
    <span class="infopiece">
        <span class="label">Price</span>
        <span class="value"><%= result.getPrice()%></span>
    </span>
</div>
<a href="main.jsp">Back to main page</a>
                </td>
            </tr>
        </table>
    </center>
</body>
</html>
```

Figure A.27 (Part2): JSP Code access the Book Information service

Appendices

In Figure A.28 Present the Front End application for accessing and consuming the POM service that exit when customer interest with any item book, if the customer order the book item this will led to consume the Core Java service which corresponded with POM Service. In this Front End application customer should be input the information details for his name, credit card number, Card Expirations, address information, quantity of item, and shipment agent. When completing all required information the customer can order this item by clicking Purchase Item to complete order.

Figure A.28: Front End Application for Ordering Item

In Figure A.29 is successful billing report response that happens. If the customer click on *Purchase* button in the Front End application of Figure A.28 will return the response of checking of POM services. if all entered information about credit card have sufficient amount of credit and expiration date are valid and quantity of item are founded in inventory will return the billing an invoice to the customer Figure A.29 depicts the successful billing report .

Figure A.29: Snapshots of Successful Response Billing Report

In Figures A.30 is Invalid billing report response that happens. If the information of credit card entered by customer is invalid or not have sufficient amount credit or the inventory not have sufficient items of book or not found in the inventory will return or response an error message as depicts in Figures A.30

Figure A.30: Snapshots of Invalid Credit Response Billing Report

Figure A.30: Snapshots of Invalid Inventory Response Billing Report

Figure A.31 depicts the JSP Code for the front end interface to access the Book Information Service

```jsp
<%@ page import= "java.util.GregorianCalendar"%>
<%@ page import="java.math.BigDecimal" %>
<%@ page import=" java.util.Date" %>
<%@ page import= "com.sun.org.apache.xerces.internal.jaxp.datatype.XMLGregorianCalendarImpl" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>BookStor Shop</title>
        <link rel="stylesheet" href="style.css" type="text/css" charset="utf-8" />
        <LINK REL="STYLESHEET" HREF="style1.css" TYPE="TEXT/CSS">
        <style type="text/css"></style>
        <%
            com.bookstorshop.BookInfo result1 = null;
            int bookID = Integer.parseInt(request.getParameter("bookId"));

            try {
                com.bookstorshop.BookInfoService service = new com.bookstorshop.BookInfoService();
                com.bookstorshop.BookInfoServiceSoap port = service.getBookInfoServiceSoap12();


                result1 = port.getBookInfo(bookID);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        %>

    </head>
    <body bgcolor="ffffff">

        <center>
            <table border="0" cellpadding="0" cellspacing="0">
                <tr>
                    <td width="800" align="center" colspan="3" background="image/navtop.jpg" height="75">
                        <i><u><b><font size="+3" color="white" face="Aquaduct"> BookStore Shop </font></b></u></i></td>
                </tr>
            </table>

            <table border="0" cellpadding="0" cellspacing="0">
                <td width="135" align="center" colspan="3" background="image/navleft.jpg"
                    style="background-repeat:no-repeat " height="328" valign="top" ></td>
                <td width="15"> </td>
                <td width="650" valign="top" colspan="3" height="328"><br>
                    <form name="Name Input Form" method="POST" action="OrderHandler">
                        <h2>Information Details</h2>
```
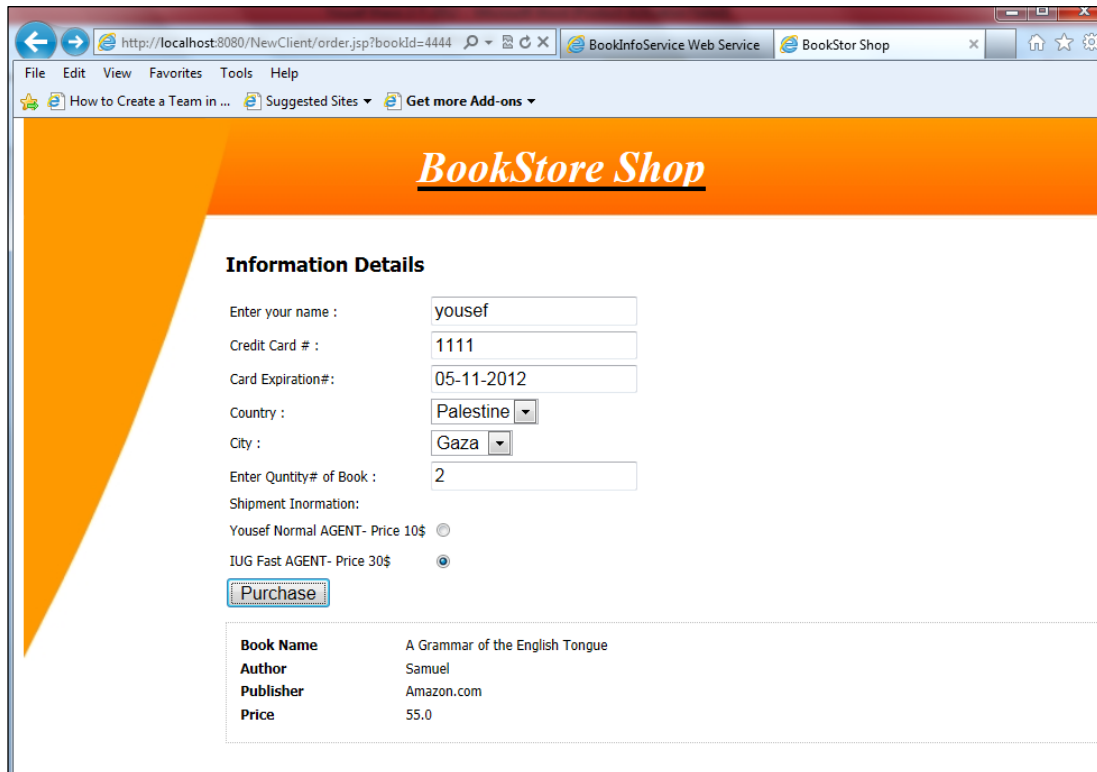
Figure A.31 (Part1): Front-End Application JSP Code for BOOKINFO.

```
<table>
    <tr>
        <td>Enter your name :</td>
        <td><input type="text" name="Name" id="Name" value="" /></td>
    </tr>
    <tr>
        <td>Credit  Card    #   :</td>
        <td><input type="text" name="CreditCardNo" value="" /></td>
    </tr>
    <tr>
        <td>Card Expiration#:</td>
        <td><input type="text" name="CardExp" value="" /></td>
    </tr>
    <tr>
        <td>Country   :</td>
        <td>
            <select name="Country">
                <option>Palestine</option>
            </select>
        </td>
    </tr>
    <tr>
        <td>City : </td>
        <td>
            <select name="City">
                <option>Gaza</option>
                <option>Rafah</option>
            </select>
        </td>
    </tr>
    <tr>
        <td>Enter Quntity# of Book :</td>
        <td><input type="text" name="Qty" value="" /></td>
    </tr>
    <tr>
        <td colspan="2">Shipment Inormation:</td>
    </tr>
    <tr>
        <td><span class="label">Yousef Normal AGENT- Price 10$</span></td>
        <td><input type="radio" name="Agent" value="Yousef" checked="checked" /></td>
    </tr>
    <tr>
        <td><span class="label">IUG Fast AGENT- Price 30$</span></td>
        <td><input type="radio" name="Agent" value="IUG" /></td>
    </tr>
</table>
```

Figure A.31 (Part2): Front-End Application JSP Code for BOOKINFO.

```
                    <input type="hidden" name="bookId" value="<%= bookID %>" />
                    <input type="submit" value="Purchase" />
            </form>
            <div class="bookinfo">

                <span class="infopiece">
                    <span class="label">Book Name</span>
                    <span class="value"><%= result1.getBookname()%></span>
                </span>

                <span class="infopiece">
                    <span class="label">Author</span>
                    <span class="value"><%= result1.getAuthor()%></span>
                </span>

                <span class="infopiece">
                    <span class="label">Publisher</span>
                    <span class="value"><%= result1.getPublisher()%></span>
                </span>

                <span class="infopiece">
                    <span class="label">Price</span>
                    <span class="value"><%= result1.getPrice()%></span>
                </span>
            </div>
        </td>
      </tr>
   </table>
  </center>
  <br><br>


  </body>    <%-- start web service invocation --%><hr/>
   <hr/>
</html>
```

Figure A.31 (Part3): Front-End Application JSP Code for BOOKINFO.

Figure A.32 depicts the JSP Code for the front end interface to access the Book Information Service

```jsp
<%@ page import= "java.util.GregorianCalendar"%>
<%@ page import="java.math.BigDecimal" %>
<%@ page import=" java.util.Date" %>
<%@ page import= "com.sun.org.apache.xerces.internal.jaxp.datatype.XMLGregorianCalendarImpl" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>BookStor Shop</title>
        <link rel="stylesheet" href="style.css" type="text/css" charset="utf-8" />
        <LINK REL="STYLESHEET" HREF="style1.css" TYPE="TEXT/CSS">
        <style type="text/css"></style>
        <%
            com.bookstorshop.BookInfo result1 = null;
            int bookID = Integer.parseInt(request.getParameter("bookId"));

            try {
                com.bookstorshop.BookInfoService service = new com.bookstorshop.BookInfoService();
                com.bookstorshop.BookInfoServiceSoap port = service.getBookInfoServiceSoap12();


                result1 = port.getBookInfo(bookID);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        %>


    </head>
    <body bgcolor="ffffff">

        <center>
            <table border="0" cellpadding="0" cellspacing="0">
                <tr>
                    <td width="800" align="center" colspan="3" background="image/navtop.jpg" height="75">
                        <i><u><b><font size="+3" color="White" face="Aquaduct"> BookStore Shop </font></b></u></i></td>
                </tr>
            </table>

            <table border="0" cellpadding="0" cellspacing="0">
                <td width="135" align="center" colspan="3" background="image/navleft.jpg"
                        style="background-repeat:no-repeat " height="328" valign="top"></td>
                <td width="15"> </td>
                <td width="650" valign="top" colspan="3" height="328"><br>
                    <form name="Name Input Form" method="POST" action="OrderHandler">
                        <h2>Information Details</h2>
                        <table>
```

Figure A.32 (Part1): Front-End JSP Code for Order Application.

```
<tr>
    <td>Enter your name :</td>
    <td><input type="text" name="Name" id="Name" value="" /></td>
</tr>
<tr>
    <td>Credit  Card    #   :</td>
    <td><input type="text" name="CreditCardNo" value="" /></td>
</tr>
<tr>
    <td>Card Expiration#:</td>
    <td><input type="text" name="CardExp" value="" /></td>
</tr>
<tr>
    <td>Country    :</td>
    <td>
        <select name="Country">
            <option>Palestine</option>
        </select>
    </td>
</tr>
<tr>
    <td>City : </td>
    <td>
        <select name="City">
            <option>Gaza</option>
            <option>Rafah</option>
        </select>
    </td>
</tr>
<tr>
    <td>Enter Quntity# of Book :</td>
    <td><input type="text" name="Qty" value="" /></td>
</tr>
<tr>
    <td colspan="2">Shipment Inormation:</td>
</tr>
<tr>
    <td><span class="label">Yousef Normal AGENT- Price 10$</span></td>
    <td><input type="radio" name="Agent" value="Yousef" checked="checked" /></td>
</tr>
<tr>
    <td><span class="label">IUG Fast AGENT- Price 30$</span></td>
    <td><input type="radio" name="Agent" value="IUG" /></td>
</tr>
</table>
<input type="hidden" name="bookId" value="<%= bookID %>" />
<input type="submit" value="Purchase" />
</form>
<div class="bookinfo">

    <span class="infopiece">
        <span class="label">Book Name</span>
```

Figure A.32 (Part2): Front-End JSP Code for Order Application

```
                <span class="infopiece">
                    <span class="label">Price</span>
                    <span class="value"><%= result1.getPrice()%></span>
                </span>
            </div>
        </td>
        </tr>
    </table>
</center>
<br><br>


</body>    <%-- start web service invocation --%><hr/>
<hr/>

</html>
```

Figure A.32 (Part3): Front-End JSP Code for Order Application.

Figure A.33 depicts the Java code Servlet Order Handler

```
package PS;

import com.bookstorshop.BookInfoService;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.ws.WebServiceRef;

import org.netbeans.j2ee.wsdl.powsdl.POService;
import org.netbeans.xml.schema.poschema.OrderInformation;
import org.netbeans.xml.schema.poschema.PaymentInformation;
import org.netbeans.xml.schema.poschema.PersonalInformation;
import org.netbeans.xml.schema.poschema.PurchaseOrder;
import org.netbeans.xml.schema.poschema.ShippingInformation;
import java.util.*;

/**
 *
 * @author yousef
 */
public class OrderHandler extends HttpServlet {
    @WebServiceRef(wsdlLocation = "WEB-INF/wsdl/localhost_54778/BookInfowS/BookInfoService.asmx.wsdl")
    private BookInfoService service_1;
    @WebServiceRef(wsdlLocation = "WEB-INF/wsdl/localhost_9080/POService/POPort.wsdl")
    private POService service;


    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
      try {
            // TODO output your page here
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet OrderHandler</title>");
            out.println("</head>");
            out.println("<body bgcolor=#ffffcc>");
            out.println("<h1><center> <font color=green size=5 >Servlet OrderHandler at "
                + request.getContextPath () + "</font></center></h1>");
```

Figure A.33 (Part1): Java Servlet Order Handler.

```
    try { // Call Web Service Operation
        org.netbeans.j2ee.wsdl.powsdl.POWSDLPortType port = service.getPOPort();
        // TODO initialize WS operation arguments here

        // ORDER INFORMATION
        OrderInformation OI=new OrderInformation();
        try { // Call Web Service Operation
            // out.println("<br/><h2>Customer Information:</h2><p></p>");
            com.bookstorshop.BookInfoServiceSoap port2 = service_1.getBookInfoServiceSoap12();
            // TODO initialize WS operation arguments here
            int bookID = Integer.parseInt(request.getParameter("bookId"));
            // TODO process result here
            com.bookstorshop.BookInfo result3 = port2.getBookInfo(bookID);
            // out.println("Result = "+result3);
            OI.setItem(result3.getBookname());
            OI.setPrice((int) result3.getPrice());

//CUSTOMER INFORMATION
        PersonalInformation PI=new PersonalInformation();
        PI.setCustomer(request.getParameter("Name"));
        PI.setAddress(request.getParameter("Country")+" -- " +request.getParameter("City"));

//PAYMENT INFORMATION
        PaymentInformation PayI=new PaymentInformation();
        PayI.setCCN(Integer.parseInt(request.getParameter("CreditCardNo")));
        PayI.setExpDate(request.getParameter("CardExp"));

//Shipment Information
    ShippingInformation SI=new ShippingInformation();
    SI.setAgent(request.getParameter("Agent"));
//PurchaseOrder
    PurchaseOrder PO=new PurchaseOrder();
    PO.setOrderInformation(OI);
    PO.setPaymentInformation(PayI);
    PO.setPersonalInformation(PI);
    PO.setShippingInformation(SI);

    // org.netbeans.xml.schema.poschema.PurchaseOrder part1 = new org.netbeans.xml.schema.poschema.PurchaseOrder();
    // TODO process result here
    org.netbeans.xml.schema.poschema.BillingReport result = port.poOperation(PO);
    // out.println("Result = "+result);
    //out.println("<div style=\"position:relative;width:400px;color:#006699;margin-left:auto;margin-right:auto;margin-top:100px;\">");
    out.println("<h1><center> <font color=#006699 size=5 >BILLING REPORT:</font></center></h1>");
    //out.println("<br/><center><h2>BILLING REPORT:</h2><p></center></p>");

if (result.getErrorReport().compareTo("NoErrors")==0)
{
    out.println("<h1><center><font color=#006699 size=1 >=================================
                                           =========================================================</font></center></h1>");

        Date date =new Date();
```

Figure A.33 (Part2): Java Servlet Order Handler.

```
        Date date =new Date();
        String months[] = {
            "Jan", "Feb", "Mar", "Apr",
            "May", "Jun", "Jul", "Aug",
            "Sep", "Oct", "Nov", "Dec"};

    int year;
    GregorianCalendar gcalendar = new GregorianCalendar();

        out.print("<table border=0 width=400px align=center bgcolor=#CDFFFF>");
        // Display current time and date information.

        out.print("Date: ");
        out.print(months[gcalendar.get(Calendar.MONTH)]);
        out.print(" " + gcalendar.get(Calendar.DATE) + " ");
        out.println(year = gcalendar.get(Calendar.YEAR));

        out.print("</tr>");
        out.println("<td></td>");
        //out.println("<td>FDGSDFGSDFG</td>");

        out.print("Time: ");
        out.print(gcalendar.get(Calendar.HOUR) + ":");
        out.print(gcalendar.get(Calendar.MINUTE) + ":");
        out.print(gcalendar.get(Calendar.SECOND));

        out.print("<td>");

        long orderId = System.currentTimeMillis();
        out.print("<center> <b> <font color=red size=3 >Order ID#:"+orderId+"</center></font></b>");

        out.print("</td>");
        out.print("</table>");
        out.print("<table border=0 width=400px align=center bgcolor=#CDFFFF>");

        out.println("<tr>");
        out.println("<td width=40%><b>Customer Name: </b></td>");
        out.println("<td width=40% align=center>"+result.getCustomer()+"</td>" );

        out.print("</tr>");
        out.println("<td>--------------------------</td>");
        out.println("<td width=40%>"+"---------------------------------------------"+"</td>" );

        out.print("</tr>");
        out.println("<td><b>Address: </b></td>");
        out.println("<td width=40% align=center>"+request.getParameter("Country")+"-"+ request.getParameter("City")+"</td>" );

        out.print("</tr>");
        out.println("<td>--------------------------</td>");
        out.println("<td width=40%>"+"---------------------------------------------"+"</td>" );
```

Figure A.33 (Part3): Java Servlet Order Handler

109

```
            out.print("</tr>");
            out.println("<td><b>Book Purchased: </b></td>");
            out.println("<td width=40% >"+result.getItem()+"</td>" );

            out.print("</tr>");
            out.println("<td><b> </b></td>");
            out.println("<td width=40%>"+"Author: "+result3.getAuthor()+"</td>" );
            out.print("</tr>");

            out.println("<td><b> </b></td>");
            out.println("<td width=40%>"+"Publisher: "+result3.getPublisher()+"</td>" );
            out.print("</tr>");
            out.println("<td><b> </b></td>");
            out.println("<td width=40%>"+"BARCODE: "+result3.getBarCode()+"</td>" );


            out.print("</tr>");
            out.println("<td>--------------------------</td>");
            out.println("<td width=40%>"+"---------------------------------------------"+"</td>" );


            out.print("</tr>");
            out.println("<td><b>Price: </b></td>");
            out.println("<td width=40% align=center>"+result.getPrice()+"$"+"</td>" );

            out.print("</tr>");
            out.println("<td><b>Quantity: </b></td>");
            out.println("<td width=40% align=center>"+result.getQty()+"</td>" );

            out.print("</tr>");
            out.println("<td><b>Shipment_Cost: </b></td>");
            out.println("<td width=60% align=center>"+result.getShippingCost()+"$"+"</td>" );

            out.print("</tr>");
            out.println("<td>--------------------------</td>");
            out.println("<td width=40%>"+"---------------------------------------------"+"</td>" );

            out.print("</tr>");
            out.println("<td><b>OverallAmount: </b></td>");
            out.println("<td width=40% bgcolor=#00FF00 align=center>"+result.getOverallAmount()+"$"+"</td>" );

            out.println("<tr>");
            out.print("</tr>");
            out.println("</table>");

            out.println("<h1><center><font color=#006699 size=1 >========================================================
                                  ============================================</font></center></h1>");
            out.println("<a href='http://localhost:8080/NewClient/'><center>Back to Main Page</center></a>");
    }

   else
```

Figure A.33 (Part4): Java Servlet Order Handler

```
   {
   out.println("<h1><center><font color=#006699 size=1 >========================================================
                                  ============================================</font></center></h1>");

   out.println("<h3 ><center>ERROR REPORT:</h3></center>");
   out.print("<table border=0 width=400px align=center bgcolor=red>");

            //out.println("<h3 ><center>Error report:</h3></center#CDFFFF>");

            out.println("<tr>");
            //out.println("<td width=40%><b>Error report: </b></td>");
            out.println("<td width=40% align=center>"+result.getErrorReport()+"</td>" );

            out.println("<tr>");
            out.print("</tr>");
            out.println("</table>");


            out.println("<h3 ><center><font color=red size=3 >Error happen May be an  Invalid Credit Card was enterd </h3></center>");

             out.println("<h3 ><center>Or </h3></center>");
              out.println("<h3 ><center>You are choose An Item that is out of inventory</h3></center>");

              out.println("<h1><center><font color=#006699 size=1 >========================================================
                                  ============================================</font></center></h1>");
              out.println("<a href='http://localhost:8080/NewClient/'><center>Back to Main Page</center></a>");

   }
            } catch (Exception ex) {
                // TODO handle custom exceptions here
                ex.printStackTrace();
            }


        } catch (Exception ex) {
            // TODO handle custom exceptions here
            ex.printStackTrace();
        }
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```

Figure A.33 (Part5): Java Servlet Order Handler

Appendix J: The Model Evaluation Scenario Based on ATAM

The ATAM relies on elicitation of quality attribute scenarios. In order to evaluate the quality of the proposed model we present in this appendix the quality attribute general scenarios for two quality attributes that are important to consider using service oriented approach which are interoperability and manageability. The questions are asked to each of the three prompts. Their answers are collected and summarized.

I. Interoperability Evaluation Scenarios

In this section we provide the general scenario related to the interoperability quality. So the evaluation of the interoperability for our proposed SOA model should consider the following scenarios concerns and their prompts are chosen for evaluation the model by which the attribute quality can be judged, specified.

1- <u>General Scenario 1</u>

Question: *Does the model have ability to use different services or systems implemented in different platforms and different programming languages.*

Prompt: The model is designed to support diverse services implemented in different platforms and languages. The model uses Web services technology which is providing interoperability. The interoperability is supported by two basic standards: SOAP and WSDL that can be used between different service providers and users, those standards communicate using standard object messaging protocol and message structure sent over web channels which is platform, programming language and communication infrastructure independent.

2- <u>General Scenario 2</u>

Question: *Does the model have the ability to use and orchestrate services implemented in different platforms and different programming languages.*

Prompt: Besides the implementing the SOAP and WSDL the model uses the BPEL in order to orchestrate Web services, which is a standard business process language that provides capabilities of the systems developed with Web service technology. The BPEL allows systems of separate underlying platforms (e.g., Java and .Net) to interact through Web services technology.

3- General Scenario 3

Question: *Does the proposed model allow having service users and providers to use different implementation languages and platforms?*

Prompt: Both service users and providers are unaware of their counter platform and a development languages, the only requirement between them is to have a unified standard for providing and invoking the service. This is achieved via WSDL, SOAP and HTTP. Moreover, in the model, we are using SOAP document-literal which is more interoperable than RPC-encoding due to incompatibility in SOAP encoding across platforms. With document-literal style web services of two section of POM agree on the exchange of complex documents that are well defined in XML schema. For example, one section sends a document describing a purchase order, the other responds with a document that describes the status of the purchase order. No need to agree on such low level details as operation names and their associated parameters. The payload of the SOAP message is an XML document that can be validated against XML schema. Document is defined by the style attribute on the SOAP binding.

4- General Scenario 4

Question: *Dose the proposed model middleware (interoperable-integrator) allow connected applications with disparate technology and data formatting requirements to interoperate as service users and providers without major changes to each?*

Prompt: The middleware interoperable or integration approach in the proposed SOA based Model will be an ESB. It is hub-and-spoke SOA approach, which provides fundamental support for Web services. Moreover the ESB allows connected applications with disparate technology, and data formatting to interoperate as service users and providers without invasive changes to each.

5- General Scenario 5

Question: *What Standards are providing interoperability to be used in the proposed model?*

Prompt: The standards used in the proposed model and provides interoperability between components and service of proposed model while interacting with each other are: XML, SOAP, WSDL, UDDI, and BPEL which provide interoperability capabilities to developed systems using Web services technology.

6- General Scenario 6

Question: *Is the interaction between a given service consumer and provider synchronous or asynchronous?*

Prompt: Services may be provided through either synchronous or asynchronous interface on SOA. In our proposed model is provides support for both synchronous and asynchronous Web service. It is left for the requirement and operation of the service used in either of them. The services implemented in the case study are synchronous services.

7- General Scenario 7

Question: *How are legacy systems are integrated-interoperated in the proposed model?*

Prompt: In SOA environment there is typically more than one reasonable way to integrate a legacy system. In our proposed model ESB the middleware integrator or interoperate is responsible for integrating legacy systems to the model, which provides interoperability with old legacy applications like direct database access. Therefore we build an interface between the legacy applications and middle were which is capable to communicate using the WSDL and SOAP, this solution will wrap the legacy application and make it act as a web service. For example the .Net service as legacy systems uses direct access database are integrated into the SOAP and WSDL which act as Web service.

8- General Scenario 8

Question: *What challenges interoperability in the proposed model?*

Prompt: the syntactic interoperability is supported by two basic standard SOAP and WSDL. There are also additional standards such as BPEL, and WS-Security and UDDI that provide capabilities to systems developed with Web services technology. But the challenges appear when not all Web services platforms implement the same versions of these additional standards where they may face some obstacles when using such standard. This risk can be mitigated when using interoperable middleware like ESB. This can utilize the using of different interfaces developed by different versions of standards

II. Manageability Evaluation Scenarios

In this section we provide the general scenario related to the manageability quality. So the evaluation of the manageability for our proposed SOA model should be consider the following scenarios concerns and their prompts are chosen for evaluation the model by which the attribute quality can be judged, specified.

1. General Scenario 1

Question: *Does the model provide strategic for exception handling and fault recovery?*

Prompt: Establishment proper of standards and exception handling which helps to detect failures. The model involve heterogeneous platforms and protocols as well as external services a robust SOA-based model must deal with system failures at service user and service provider, data services. Error handling strategies should manage the behavior of the model under failure modes.

2. General Scenario 2

Question: *Does the model provide access point for managing Web service and manageability access information?*

Prompt: the ESB provides access point for managing Web services. The management service of the model is to achieve a manageability endpoint for managing a Web service. By utilizing manageability, consumer should get the manageability endpoint precisely through simple Web service interface.

3. General Scenario 3

Question: *Does BPEL process and environment provide support for monitoring and logging event data to allow the measurement of business metric?*

Prompt: The BPEL engine manages BPEL processes and the application server that runs the engine in its context provides monitoring and logging of event data and measurement of business metrics such as wait time, transaction volumes, and exception counts.

4. General Scenario 4

Question: *Does each of the implemented BPEL processes properly deal with business and technical exception conditions*

114

Prompt: Services in the business process can be properly managed and can deal with business and technical exception condition when using the composite services in the BPEL workflow.

5. <u>General Scenario 5</u>

Question: *Does the model support monitoring facility?*

Prompt: The model through the ESB and application server provides a monitoring facility for the invoked services, and the health of the model components.

6. <u>General Scenario 6</u>

Question: *Does the model support stop/start components and services?*

Prompt: The ESB and the application server under which the services run allow to start and stop model engines, components, and services.