

**FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY  
UNIVERSITY OF MALAYA**



# **WEB SERVICES FOR E-PROCUREMENT SYSTEM IN THE CONTEXT OF SUPPLY CHAIN MANAGEMENT**

**A thesis submitted to  
the Faculty of Computer Science and Information Technology,  
University of Malaya.**

**by**

**YAP KHAY SENG**

**WGC020054**

*Supervisor:*

**PUAN SITI HAFIZAH BINTI AB. HAMID**

**Session 2006/2007**

**In partial fulfillment of the requirement for the Degree  
of Masters in Software Engineering, University of Malaya**

**WEB SERVICES FOR E-PROCUREMENT  
SYSTEM IN THE CONTEXT OF SUPPLY  
CHAIN MANAGEMENT**

**YAP KHAY SENG  
WGC020054**

**FACULTY OF COMPUTER SCIENCE  
AND INFORMATION TECHNOLOGY  
UNIVERSITY OF MALAYA  
SESSION 2006/2007**

## **ABSTRACT**

The current Information Technologies (IT) such as Electronic Data Interchange (EDI) and Enterprise Application Integration (EAI) have been used by major organizations to integrate the business processes in the supply chain. However, these technologies are expensive, inflexible and not dynamic. Web Services seem to be a way to solve these problems. By using the standard internet protocol, Web Services can easily provide interoperable software service functions that can be accessible by various hardware and platform, even passing through the firewalls. The purpose of this project is to develop a system using Web Services to implement real time information sharing and dynamic procurement operations in the context of Supply Chain Management. Web Services for E-Procurement (WSEP) System is a web-based procurement system that provide real time knowledge sharing and procurement operations. The system is deployed using Web Services which is not only easier to implement but also integrated well with legacy system. The system is running under the Microsoft .NET 1.1 Framework and Microsoft SQL Server 2000 as the backend database.

## **ACKNOWLEDGEMENTS**

I would like to acknowledge many people for helping me during this project. First, I would especially like to thank my project supervisor, Ms Siti Hafizah, for her generous time and commitment. Her invaluable comments and guidance have inspired me to complete this project within the time schedule.

I extend many thanks to my friends, especially Hu Heng Siew, Marina and Tan Chin Kuang, who have provided feedback and inspiration in one way or another that contributed to the success of this project. Their knowledge of Web Services was also extremely helpful.

Finally, I would like to thank my parents and sisters for their unwavering love. Without their support and encouragement, I would never have been able to finish this project. I am especially grateful to my wife Min Min for her patience and for helping me keep my life in proper perspective and balance.

<b>TABLE OF CONTENTS</b>	<b>Page</b>
Abstract	i
Acknowledgements	ii
Table of Contents	iii
List of Figures	viii
List of Tables	xi
Abbreviations	xii
 <b>CHAPTER 1: INTRODUCTION</b>	
1.1 Introduction	1
1.2 Current Problems	2
1.3 Project Objectives	3
1.4 Project Scope	3
1.5 Project Limitation	4
1.6 End User	4
1.7 Expected Outcome	5
1.8 Significance of Project	5
1.9 Research Methodology	5
1.10 Project Schedule	6
1.11 Chapters Summary	8
 <b>CHAPTER 2: LITERATURE REVIEW</b>	
2.1 Supply Chain	9
2.2 Supply Chain Management (SCM)	10
2.3 E-Procurement	11
2.3.1 Overview of Purchasing Processes	13
2.3.2 Advantages of E-Procurement	14

2.3.3	Current Issues and Problems of Information Technologies (IT) in the Context of Supply Chain Management (SCM)	16
2.4	Web Services	17
2.4.1	Introduction	17
2.4.2	Definition of Web Service	18
2.4.3	Web Service Technologies	19
2.4.3.1.	eXtended Markup Language (XML) and XML Schema	21
2.4.3.2.	Simple Object Access Protocol (SOAP)	23
2.4.3.3.	Web Services Description Language (WSDL)	23
2.4.3.4.	Universal Description, Discovery, and Integration (UDDI)	24
2.4.3.5.	Microsoft .NET Framework	26
2.5	Existing Project	29
2.5.1	Electronic Business using eXtensible Markup Language (ebXML)	29
2.5.2	Web Service Intelligent Agent Structuring for Supply Chain Management	32
2.5.3	Web Service-based Framework for Supply Chain Management	33
2.5.4	Comparison of the Proposed Framework With Others (ebXML, Umair and Ben, Zhang Mi et al.)	36
2.6	Conclusion	36

### **CHAPTER 3: METHODOLOGY**

3.1	Research Methodology	37
3.2	Development Methodology	39
3.2.1	Choice of Modified Waterfall Model	39
3.2.1.1	Phase 1 – Requirements	40
3.2.1.2	Phase 2 – Design	40
3.2.1.3	Phase 3 – Implementation	41
3.2.1.4	Phase 4 – Testing	41
3.3	Techniques Used to Define Requirements	41
3.2.1	Analysis of existing projects	42
3.2.2	Review of new technologies	42

3.2.3	Library Research and Internet Research	42
-------	--	----

## **CHAPTER 4: SYSTEM REQUIREMENT ANALYSIS**

4.1	Functional Requirements	43
4.1.1	Use Case Diagram	44
4.2	Non-Functional Requirements	55
4.3	Decisions on the choices of Development Technologies	56
4.3.1	Web Services Development Platform	56
4.3.2	Web Server	58
4.3.3	Database Server	58
4.3.4	Development Language	58
4.4	Hardware Requirements	58

## **CHAPTER 5: SYSTEM DESIGN**

5.1	Web Services E-Procurement Architecture Design	62
5.1.1	Authentication Module	64
5.1.2	Maintenance Module	64
5.1.3	E-Procurement Transaction Web Services Module	64
5.1.4	Suppliers' Web Services	64
5.2	Web Services E-Procurement System Functionality Design	65
5.3	The Design of Dynamic Aspect of WSEP System	75
5.4	Web Services E-Procurement System Design	76
5.5	Web Services E-Procurement Components Design	79
5.6	Web Services E-Procurement Database Design	82
5.6.1	Database Diagram	82
5.6.2	Data Dictionary	83
5.7	User Interface Design	85
5.8	Conclusion	87

## **CHAPTER 6: SYSTEM IMPLEMENTATION**

6.1	Development Environment	88
-----	-------------------------	----

6.1.1	Software Development Environment	88
6.1.2	Hardware Development Environment	88
6.2	System Implementation	89
6.2.1	Implementation of Data Access Tier	89
6.2.2	Implementation of Business Logic Tier	91
6.2.3	Implementation of the Web Services	92
6.2.4	Implementation of WSEP Database	98
6.2.5	Implementation of User Interface using ASP.NET	99
6.2.6	Implementation of Supplier Web Services	102
6.3	Implementation of Dynamic Aspect of WSEP System	103
6.4	Conclusion	107

## **CHAPTER 7: SYSTEM TESTING**

7.1	Testing Approach	108
7.1.1	Unit Testing	109
7.1.1.1	Black Box testing	109
7.1.1.2	White Box Testing	111
7.1.2	Module Testing	113
7.1.3	Integration Testing	115
7.1.4	System Testing	115
7.2	Non-Functional Requirements Testing	116
7.3	WSEP Metrics	123
7.3.1	Coupling	124
7.3.2	Cohesion	125
7.3.3	Analysis of Result	127
7.4	Conclusion	129

## **CHAPTER 8: CONCLUSION**

8.1	Problems Encountered	130
-----	----------------------	-----



8.2	System Strengths	131
8.3	System Constraints and Future Enhancement	132
8.4	Knowledge and Experience Gained	133
8.5	Conclusion	135

## **References**

## **Appendices**

- A. WSDL of Supplier Web Services
- B. User Manual

<b>List of Figures</b>	<b>page</b>
<i>Figure 1.1 Project Time Line</i>	7
<i>Figure 2.1 The publish, find, and bind paradigm of Web Services</i>	20
<i>Figure 2.2 How UDDI works</i>	25
<i>Figure 2.3: The .NET Framework architectural components</i>	27
<i>Figure 2.4: ebXML System Overview</i>	30
<i>Figure 2.5 Proposed Web Service Agent by Umair and Ben</i>	32
<i>Figure 2.6 Framework of Supply Chain Management proposed by Zhang Mi et al.</i>	35
<i>Figure 3.1 Research Methodology</i>	38
<i>Figure 3.2 Modified Waterfall Model with iterative development</i>	40
<i>Figure 4.1 Use Case Diagram of WSEP System</i>	45
<i>Figure 5.1 Proposed Architecture of Web Services E-Procurement (WSEP) System</i>	63
<i>Figure 5.2: Sequence Diagram for User Login</i>	65
<i>Figure 5.3 Sequence Diagram for List Suppliers</i>	66
<i>Figure 5.4 Sequence Diagram for Add Supplier</i>	67
<i>Figure 5.5 Sequence Diagram for Edit Supplier</i>	67
<i>Figure 5.6 Sequence Diagram for Delete Supplier</i>	68
<i>Figure 5.7 Sequence Diagram for List Products</i>	69
<i>Figure 5.8 Sequence Diagram for Add Product</i>	69
<i>Figure 5.9 Sequence Diagram for Edit Product</i>	70
<i>Figure 5.10 Sequence Diagram for Delete Product</i>	71
<i>Figure 5.11 Sequence Diagram for List Supplier Web Services URI</i>	71
<i>Figure 5.12 Sequence Diagram for Add Supplier Web Services URI</i>	72
<i>Figure 5.13 Sequence Diagram for Edit Supplier Web Services URI</i>	73
<i>Figure 5.14 Sequence Diagram for Delete Supplier Web Services URI</i>	73
<i>Figure 5.15 Sequence Diagram for Request for Quotation</i>	74
<i>Figure 5.16 Sequence Diagram for Raise Purchase Order</i>	75
<i>Figure 5.17 The n-Tier Architecture of the WSEP System</i>	77

<i>Figure 5.18 Elements of the Data Tier of Microsoft SQL 2000 Server</i>	78
<i>Figure 5.19 Deployment Diagram for BLLeProcurement</i>	80
<i>Figure 5.20 Deployment Diagram for DLLeProcurement</i>	81
<i>Figure 5.21 Deployment Diagram for DynamicWebServicesLib</i>	81
<i>Figure 5.22 Database design of the WSEP System</i>	83
<i>Figure 5.23 Screen uses terms and objects that have direct analogues in the user's environment and familiar to the user</i>	86
<i>Figure 5.24 Confirmation of destructive actions</i>	87
<i>Figure 6.1 The n-Tier Architecture of the WSEP System</i>	89
<i>Figure 6.2 Function GetAll() that returns a Dataset in the Data Access Tier component</i>	91
<i>Figure 6.3 Function GetAll() in the Business Logic Tier component</i>	92
<i>Figure 6.4 List of all web methods in Web Services</i>	93
<i>Figure 6.5 Web Method GetVendor() in the Web Services component</i>	94
<i>Figure 6.6 Web Method RequestForQuotation () in the Web Services component</i>	97
<i>Figure 6.7 Create Table and Relationship using SQL Server Enterprise Manager</i>	98
<i>Figure 6.8 Detailed View Database Diagram</i>	99
<i>Figure 6.9 Adding Web Reference through Visual Studio 2003</i>	100
<i>Figure 6.10 Web Services on the Local Machine</i>	100
<i>Figure 6.11 Dialog box to Add Reference</i>	101
<i>Figure 6.12 WSDL of the Supplier Web Services</i>	103
<i>Figure 6.13 Sample of implementation of Supplier Web Services</i>	104
<i>Figure 6.14: Generate Code From Namespace</i>	105
<i>Figure 6.15: Compile the Codes to Assembly</i>	105
<i>Figure 6.16: Create Instance from Assembly using Activator Class</i>	106
<i>Figure 6.17: Public Function InvokeCall() in DynamicWebServicesProxyLib</i>	106
<i>Figure 7.1 Testing Process</i>	109
<i>Figure 7.2 Black Box Testing of a Web Service</i>	110

<i>Figure 7.3 Successful Output of Black Box Testing</i>	<i>111</i>
<i>Figure 7.4 White Box Testing Examining and analyzing the code structure</i>	<i>113</i>
<i>Figure 7.5 Units in a Vendor Module</i>	<i>115</i>
<i>Figure 7.6 Assemblies that are Created Dynamically</i>	<i>117</i>
<i>Figure 7.7 Visual Studio .NET Object Browser Is Used to Examine the Dynamic Assembly</i>	<i>117</i>
<i>Figure 7.8 Windows Forms Performance Test Screen</i>	<i>118</i>
<i>Figure 7.9 Algorithm for Performance Testing of Request For Quotation Transaction</i>	<i>119</i>
<i>Figure 7.10 Bar Chart for Test Result of CPU usage and Memory usage</i>	<i>120</i>
<i>Figure 7.12 Bar Chart for Test Result of SQL Response Time</i>	<i>120</i>
<i>Figure 7.12 SQL Profiler for SQL Response Time Testing</i>	<i>121</i>
<i>Figure 7.13 Windows Performance Utility Tool for Memory and CPU Usage Testing</i>	<i>122</i>
<i>Figure 7.14: Software Quality Model</i>	<i>124</i>
<i>Figure 7.15: Bar Char for the WSEP Metrics Result</i>	<i>128</i>

**List of Tables****page**

<i>Table 2.1 Different terms used for acquiring external products or services</i>	<i>12</i>
<i>Table 2.2 Compilation list of purchasing process by Knudsen</i>	<i>13</i>
<i>Table 4.1 Comparison of .NET and Java Platforms</i>	<i>57</i>
<i>Table 4.2 The minimum requirements to install the .NET Framework 1.1 SDK</i>	<i>59</i>
<i>Table 4.3 The minimum requirements to run Web Services on the .NET Framework</i>	<i>60</i>
<i>Table 5.1 Table Structure of WSEP System</i>	<i>84</i>
<i>Table 5.2 User interface design principles</i>	<i>85</i>
<i>Table 7.1: Test Result of Performance</i>	<i>119</i>
<i>Table 7.2: Results of Each Module Being Analyzed</i>	<i>128</i>

## Abbreviations

.NET	Microsoft .NET Framework
ADO.NET	Microsoft ActiveX Data Objects used in the .NET Environment
API	Application Programming Interface
ASP.NET	Microsoft Active Server Pages used in the .NET Environment
B2B	Business To Business
BPSS	Business Process Specification Schema
CLR	Common Language Runtime
CPA	Protocol Agreement
CPP	Collaboration Protocol Profiles
DTD	Document Type Declaration
ebMS	ebXML Messaging Service
ebXML	Electronic Business using eXtensible Markup Language
EDI	Electronic Document Interchange
ERP	Enterprise Resource Planning
HTTP	Hypertext Transport Protocol
IDE	Integrated Development Environment
IIS	Microsoft Internet Information Service
ISP	Internet Service Provider
IBM	International Business Systems
J2EE	Java 2 Enterprise Edition
JIT	Just-In-Time
JRE	Java Runtime Engine

MSDN	Microsoft Developer Network
MSIL	Microsoft Intermediate Language
OASIS	Organization for Advancement of Structured Information Standards
RPC	Remote Procedure Call
SCM	Supply Chain Management
SDK	Software Development Kit
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Standard Query Language, the de facto standard language of all databases
TCP/IP	Transport Control Protocol / Internet Protocol
UDDI	Universal Description, Discovery and Integration
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
URI	Uniform Resource Identifier
VB.NET	Visual Basic.NET
W3C	World Wide Web Consortium
WSEP	Web Services For E-Procurement System
WSDL	Web Service Description Language
XML	eXtensible Markup Language
XSD	XML Schema

## **CHAPTER 1: INTRODUCTION**

### **1.1 Introduction**

In an increasingly competitive marketplace today, producing quality products is not enough to meet customers' expectations and requirements. Customers are now demanding the same levels of quality not only in the product itself, but also in the services that are offered with it. Looking for means to respond to these requirements, enterprises have to reorganize their business processes from traditional operations toward integrated and strategic partnerships. The result was the emergence of the concept of Supply Chain Management (SCM).

“A supply chain is a network of facilities that procure raw materials, transform them into intermediate goods and then final products, and deliver the products to customers through a distribution system” (Lee and Billington, 1995). By uniting the disparate business processes, enterprises in the supply chains can effectively improve transaction performance and reduce costs. In the context of procurement, extending the business processes across the enterprise involves collaboration of suppliers, which includes the activities of involvement in production design, daily coordination of purchased material flows and etc. These activities require system integration as well as information sharing in a real time manner. However, current supply chain implementations in Electronic Data Interchange (EDI) systems do not really meet these requirements.

Web Services are the consequence of the evolution of the World Wide Web (WWW) that provide a standard protocols and data formats for implementing distributed components. By using ubiquitous and standard internet protocol, Web services can easily provide interoperable software service functions that can be accessible by various hardware and platform, even passing through the firewalls. Therefore, enterprises can use Web Services to expose their business processes to the external trading partners. The advent of Web Services has simplified Business-To-Business (B2B) integration.



The purpose of this research project is to design and develop a system, Web Services for E-Procurement System (WSEP), using Web Services to apply dynamic procurement operations and real time information sharing. The system is implemented using Microsoft .NET framework that provides build-in functions and supports for Web Services. The architecture and design of the system presented in this project are based on an n-tier enterprise design pattern.

## **1.2 Current Problems**

Some of the enterprises nowadays still rely on manual processes to identify, evaluate and select suppliers. This practice is expensive to maintain and prone to human errors. Besides, these manual processes do not accommodate future growth and cannot consistently produce the level of services that is needed to optimize the procurement processes. With the implementation of Supply Chain Management (SCM), the organizations can streamline the business processes, improve response time to customer requests and reduce costs. SCM coordinates the processes in production management, inventory management, order fulfillment, purchasing and distribution.

A Supply Chain typically extends the business process across and beyond organization. Information is passed through different systems of the companies across the supply chains. Electronic Data Interchange (EDI) and Enterprise Application Integration (EAI) have been used by major organizations to integrate the business processes in the Supply Chain. However, these technologies are expensive, inflexible and not dynamic. If some of the participating systems change, the integration must be redone which is a time consuming and expensive process (Zhang Mi, 2005). These problems hinder the applications of dynamic SCM.

Web Services seem to be a way to solve these problems. This project uses Web Services technologies to overcome such problems. Web Services for E-Procurement System (WSEP) is a web-based E-Procurement system that demonstrates the use of Web Services to provide dynamic procurement operations and real time information sharing

over the internet.

### 1.3 Project Objectives

The objectives of this project are:

1. To design an n-tier architecture using Web Services to apply real time and dynamic procurement operations in the context of Supply Chain Management.
2. To implement the project architecture running on .NET framework.
3. To evaluate the system by using software testing and software metrics.

### 1.4 Project Scope

This project will cover the design, development, testing as well as evaluation of Web Services for E-Procurement (WSEP) System that includes the following modules:

- **Authentication Module**

This module is to authenticate and authorize the user. This involves accepting credentials from the users that include username and password, and validating them against the database. Only purchase officers with access permission are allowed to access to the system.

- **Maintenance Module**

This module is to manage and maintain the system's entities such as products, suppliers and suppliers' Web Services Uniform Resource Identifier (URI).

- **E-Procurement Transaction Web Services Module**

This module is the main component that handles the interactions between the E-Procurement system and the suppliers' Web Services. It queries the database in order to get the qualified suppliers information as well as suppliers' Web Services URI. Based on the information, it will create a proxy dynamically to bind the suppliers' Web Services for procurement operations.

- **Suppliers' Web Services Module**

The Suppliers' Web Services are necessary as part of the WSEP System. All the suppliers who want to participate in the system have to implement Web Services that conform to the predefined Web Services Description Language (WSDL) standard, which is shown in Appendix A. These Web Services integrate with the supplier internal system to expose the product and inventory information that enables the submission the quotation in real time.

### **1.5 Project Limitation**

The WSEP System does not really support sophisticated business processes. The system assumes that the purchasing processes and transactions are straight forward. However, the system could be extended to support more complicated business processes.

Due to the limitation of time, this project does not address Web Services Security. Security is one of the major concerns of many potential Web Services adopters. Web Services do not have widely adopted security protocols and standards dedicated for Web Services. This project uses existing security measures such as HyperText Transport Protocol Secure (HTTPS) to encrypt the whole Simple Object Access Protocol (SOAP) messages to secure Web Services communications.

### **1.6 End user**

The system will be mainly used by purchase officer to get product pricing and to check product availability from the suppliers' inventory system. By using the Web Services, the suppliers who participate in the WSEP system can provide information and accept purchase order placed by the purchase officer.

### **1.7 Expected Outcome**

The system is designed to be functioned in real time and dynamic manner. Based on the decision made by the purchase officer, the system will dynamically bind the suppliers' Web Services to retrieve the information needed, such as product pricing and inventory level and process the transactions. The return results are shown to the user.

### **1.8 Significance of The Project**

Web Services provide a standard-based approach to implement distributed components. They offer data and business logic services over standard protocols namely HyperText Transport Protocol (HTTP), eXtensible Markup Language (XML), and Simple Object Access Protocol (SOAP) over the Internet. Using the ubiquitous and low-cost Internet, Web Services can easily provide interoperable software functions over the Intranet and the Internet. By deploying Web services, this project significantly reduces the integration costs of Supply Chain Management, particularly on E-Procurement application. Besides, by using the dynamic nature of Web Services, this project shows that the E-Procurement process can be done in real time and dynamic manner. Information sharing from the suppliers can be easily become available to the system.

The development of this project uses technologies namely Web Services and .NET Framework that are supposed to improve upon older technologies such as EDI in terms of easy of use of development, scalability and maintainability.

### **1.9 Research Methodology**

The research methodology that is used for this dissertation consisted of literature review and research findings. The data was collected and analyzed on the research materials such as published papers, journals, white papers, books and other documents. The gathered requirements are then modeled and captured using use case diagrams.

The system is built based on Microsoft .NET XML Web Services technology. Web Services provide efficient application integration independent of platforms and hardware. The dynamic binding aspect of the Web Services are examined and applied to the system. This includes creating a proxy to invoke the Web Services at run time using Microsoft .NET framework.

Object-oriented metrics is used in this project to assess the quality aspects of the system such as maintainability and reliability. The coupling and cohesion of the system design are measured and analyzed to evaluate the overall quality of the system.

### **1.10 Project Schedule**

The project schedule spans from Feb 2005 to June 2006. The Gantt Chart in Figure 1.1 shows the tasks and schedules of all phases of the project.

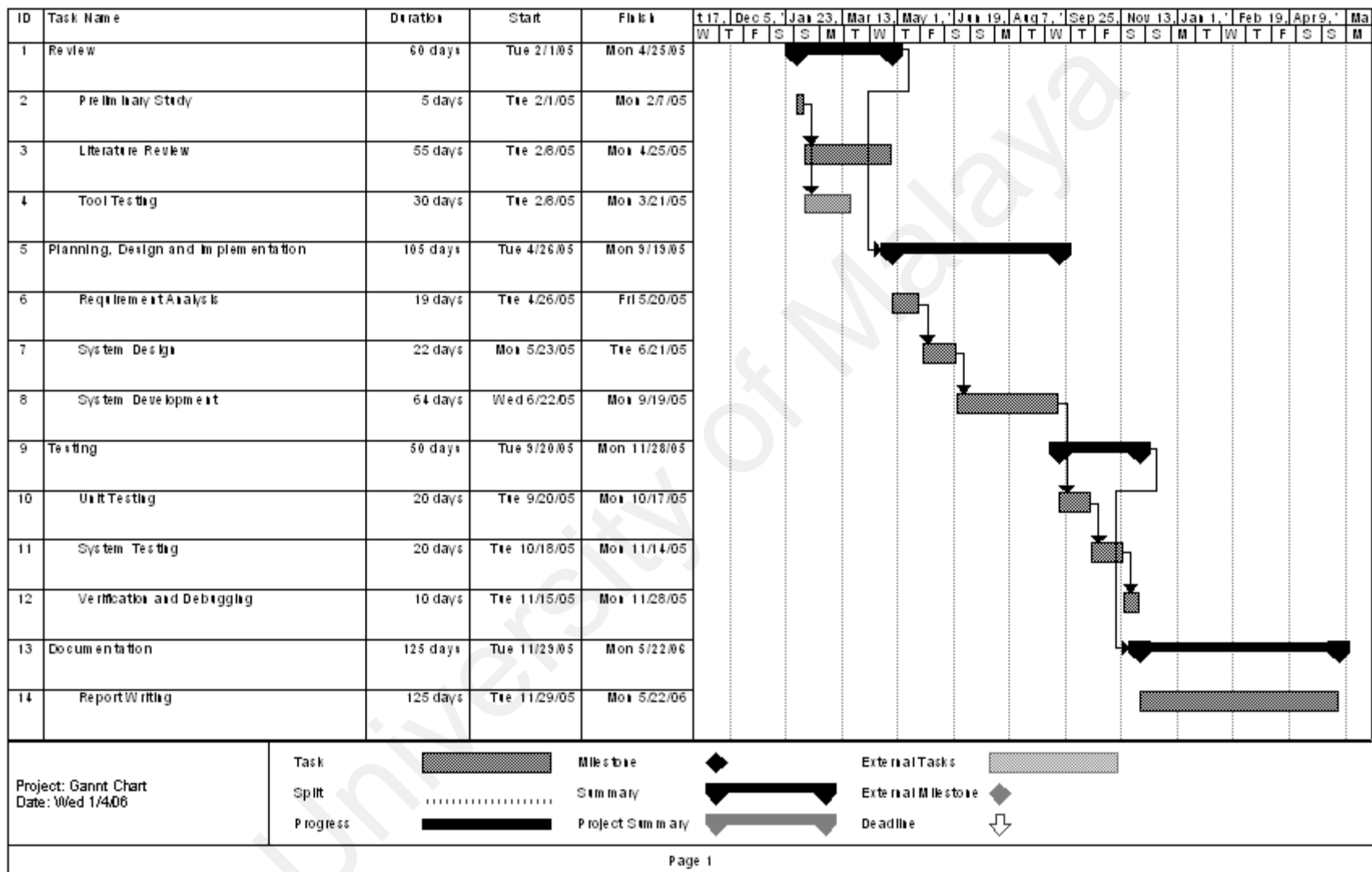


Figure 1.1 Project Time Line

## **1.11 Chapters Summary**

This is a brief explanation of the chapters included in this report to give an overview of what is contained herein.

Chapter 1: “Introduction” gives an overview of the project, current problems, project objectives, significance of the project, project scope, project limitations as well as the project schedule.

Chapter 2: “Literature Review” is a study on existing systems and projects. The study includes all the development paths leading to the core technologies used in the project.

Chapter 3: “Methodology” discusses the research and development methodology used in arriving at this system, as well as the techniques used to define the requirements.

Chapter 4: “System Requirement Analysis” lays out the details of system requirements, including functional and non-functional requirements, hardware and software requirements. There is also a comparison of the various development platforms and languages reviewed.

Chapter 5: “System Design” shows in detail the system design, database design and user interface design.

Chapter 6: “System Implementation” describes the development environment and strategy.

Chapter 7: “System Testing” shows various testing methods and software metrics that are used to validate and measure different aspects of the system.

Chapter 8: “Conclusion” discusses problems encountered, system strengths, system constraints, future enhancements, and knowledge and experience gained.

## **CHAPTER 2: LITERATURE REVIEW**

This chapter covers the study on existing projects and frameworks. The study seeks to discover the knowledge and ideas that have been established on Supply Chain Management, E-Procurement and Web Services in terms of their strengths and weaknesses.

### **2.1 Supply Chain**

Jayashankar (Janyashankar et al, 1996) defines a supply chain to be “a network of autonomous or semi-autonomous business entities collectively responsible for procurement, manufacturing, and distribution activities associated with one or more families of related products.”

Lee and Billington (Lee and Billington, 1995) has a similar definition: “A supply chain is a network of facilities that procure raw materials, transform them into intermediate goods and then final products, and deliver the products to customers through a distribution system.”

The supply chain represents the flow of materials, information and funds as they move in a process from supplier to manufacturer to wholesaler to retailer and to consumer. The supply chain activities transform raw materials and components into a finished product that is delivered to the end customer. The elements of a supply chain typically consist of production planning, material sourcing, transportation management, warehouse management and demand management. These functions are tightly integrated to provide the products and services to the end user in an efficient, timely and profitable manner (Scott and Oldfield, 2004).

In addition to internal functions, the supply chain also comprises the activities of external entities which include materials and parts suppliers, manufacturers, distributors, and transportation providers. The supply chain not only encompasses the movement of goods between supply chain participants, but also the flow of information and funds.



Supply Chain execution begins at the point a demand is created and is about the efficiency and effectiveness with which that demand is fulfilled (Scott and Oldfield, 2004). Many organizations are looking to supply chain optimization as a means of gaining significant competitive advantages.

## **2.2 Supply Chain Management (SCM)**

Global competition is forcing enterprises to restructure their business process from conventional operation towards integrated partnership. One of the new strategic management philosophies that serves to integrate a number of best practices is the area of Supply Chain Management (SCM) (Ines, 2002). By integrating the disparate business processes in the supply chain, the enterprises experience noteworthy performance improvements in transaction and production related costs, in asset utilization, and in responsiveness to customers needs.

SCM was defined as “the integration of business processes from end user through original suppliers, that provides products, services and information that add value for customers and other stakeholders” (Cooper, 1997) .

According to (Christopher, 1992), “Supply Chain Management is the process of strategically movement and storage of materials, parts and finished inventory from supplier through the firm and on to the customer.”

Ellram and Cooper (Ellram and Cooper, 1990) defined SCM as “an integrative philosophy to manage the total flow of a distribution channel from the supplier to the ultimate user.”

From the above definitions, SCM can be summarized as “the management of materials and information flows both in and between facilities across Supply Chain.” SCM is a cross-functional approach in managing the movement of raw materials into an organization and the movement of finished goods out of the organization toward the end-

consumer. As organizations are striving to focus on core competencies and become more flexible, they have reduced their ownership of raw materials sources and distribution channels. These functions are increasingly being outsourced to other corporations that can perform the activities better or more cost effectively. The effect has been to increase the number of companies involved in satisfying consumer demand, while reducing management control of daily logistics operations. Less control and more supply chain partners led to the creation of SCM concepts. The purpose of SCM is to improve trust and collaboration among supply chain partners and thus increasing efficiency and profitability.

### **2.3 E-Procurement**

One of the key supply chain activities is procurement. This activity is one of the key factors that will determine the success of the enterprise. E-Procurement, by using e-business mechanisms deployed on procurement, offers many innovative methods that can improve the process of procurement. E-Procurement systems can improve efficiency and effectiveness by automating processes, replacing human labor with information technology, facilitating the breakdown of functional silos towards horizontal processes that cut across departments and divisions (Neef, 2001).

In order to define the term “E-Procurement”, we need to know the meaning of Procurement and Electronic Business (E-Business). The terms ‘procurement’, ‘purchasing’ and ‘supply management’ are always used interchangeably for the meaning of acquiring external products or services for an organization. Table 2.1 shows the different terms and meaning respectively proposed by Knudsen (Knudsen, 1999).

*Table 2.1: Different terms used for acquiring external products or services  
(Knudsen, 1999)*

Term	Meaning
Supply Management	To be aware of the strategic impact of procurement and fully exploit it by formulating a supply strategy
Procurement	To satisfy internal demands with external sources which adhere to objectives set at the strategic level
Purchasing	The minimum activities required to obtain external products or services that result in an invoice from an external source

Min and Galle state that Electronic Business generally refers to “an inter-organizational information system that is intended to facilitate Business-To-Business (B2B) electronic communication, information exchange and transaction support through a web of either public access or private value-added networks” (Min and Galle, 1999).

Amit and Zott define that business conducted over the internet is E-Business (Amit and Zott, 2001).

IBM defines e-procurement as “the acquisition of direct and indirect products and services using the Internet and new technologies to facilitate a seamless, end-to-end stream of strategic procurement activities by connecting buyers with suppliers. Typically includes tools and business intelligence systems that enable improved responsiveness and analysis within the procurement organization” (IBM, 2005).

By combining the procurement and e-business definition, we can define E-Procurement as “the use of IT and Internet to satisfy internal demands with external sources which adhere to objectives set at the strategic level”.

### 2.3.1 Overview of Purchasing Processes

Knudsen has compiled a list of activities of purchasing process done by Novack and Simco (1991), Archer and Yuan (2000) and van Weele (2002). Table 2.2 shows the compilation list of purchasing processes.

*Table 2.2 Compilation list of purchasing process by Knudsen (Knudsen, 2003)*

Novack and Simco	Archer and Yuan	van Weele
Identify or re-evaluate needs	Information gathering (Search for suppliers that can satisfy requirements)	Determining the specification of goods and services that need to be bought
Define and evaluate user requirements		
Identify type of purchase		
Conduct market analysis		
Identify all possible suppliers	Contact supplier	
Prescreen all possible supplier	Do background review	
Evaluate remaining supplier base		Identify most suitable supplier
Choose Supplier	Negotiate contract	Preparing and conducting negotiations
Deliver product/ performance service	Fulfillment (shipment, delivery and payment)	Placing an order with the selected supplier
Post-purchase / make performance evaluation	Consumption, maintenance and disposal (evaluation of performance)	Monitor and control the order
	Renewal (when product is consumed)	After-care and evaluation

From the above table, the purchasing process can be summarized of having 4 basic steps:

**1. Identify and determine needs**

The procurement process starts when an organization's needs have been evaluated and identified. The specification of goods and services that need to be bought is determined.

**2. Evaluate and Select suppliers**

All possible suppliers are identified. Suppliers are assessed and evaluated based on various dimension such as quality, delivery, price, supports and etc. Buying organization negotiates with potential suppliers regarding price, delivery date and other variables. One or more suppliers will be selected.

**3. Place order**

Orders are placed with the selected and qualified suppliers. Products or services are delivered to the buying organization. Orders are fulfilled.

**4. Post-purchase and evaluation of performance**

Post purchase activities include monitoring and evaluating suppliers' performance and capabilities. This is to ensure that the suppliers' deliveries fulfill the specified requirements.

**2.3.2 Advantages of E-Procurement**

In the field of E-Business, E-Procurement is regarded as having far greater potential for cost savings and business improvements than online retailing or Enterprise Resource Planning (ERP) systems (Neef, 2001). The benefits come not only through direct cost savings but also through the improved operational efficiency of companies. They can shrink dramatically the number of suppliers with whom they deal, reduce the administration costs and gain a clearer picture of their overall purchasing strategy. For large corporations in particular, E-Procurement may even be the most important element

of E-Business for operational excellence (Barua et al., 2001).

E-Procurement is the automation of the procurement processes so that the sourcing, vendor selection, purchasing processes, shipment status tracking and payments can be being made through the internet. With the increasing pressures on global competition, organizations are looking at various ways to reduce the costs. This has given a different definition to the way procurement has been functioning, and making it the strategic section for cost reduction. The main purpose of the E-Procurement is to reduce the cycle time for executing purchasing processes which directly drive down the costs (Knudsen, 2003). Besides, E-Procurement has increased the opportunities to attract greater number of suppliers. This also provides for greater bargaining power to the buyers and leads to huge cost advantage to the organization. In addition, E-Procurement gives a sharper view of disparities which occur between pricing, quality and delivery among the suppliers.

With the advent of internet, companies can communicate and interact with their suppliers, customers and employees to an extent that never before possible. Thus, E-Procurement does not only mean putting purchasing decisions online, but also means linking suppliers into the purchasing network and broadening the range of employees who can carry out transactions. This enables aggregation of purchasing process across multiple departments or divisions without removing individual control. In other words, E-Procurement improves the operational efficiency of the company (Knudsen, 2003).

From perspective of the suppliers, E-Procurement enables the suppliers to become more proactive in the way that they do business. Instead of just showing the products in a catalogue and waiting to be approached by buyers, the suppliers can be linked into companies' inventory systems to see when goods may be due for renewal. Indirectly, this has created a more reactive purchasing policy for the buyers to find the best price and quality across a wide range of suppliers.

### **2.3.3 Current Issues and Problems of Information Technologies (IT) in the Context of Supply Chain Management (SCM)**

A supply chain typically involves multiple parties, including manufacturers, transportations, warehouses, retailers as well as the customers. Information Technology (IT) has been playing a key role in improving the efficiency of the supply chain by integrating information system across multiple organizations.

For more than two decades, enterprises have been using Electronic Data Interchange (EDI) systems for basic B2B document exchange, i.e. orders, invoices, shipment notices and etc. EDI enables a company's trading partners to transmit requests or documents directly into their business systems. For instance, an electronic version of a purchase order can be transmitted from the buyer and delivered directly into the order processing system of the seller. EDI translation software converts the incoming electronic purchase order from a commonly understood format into the proprietary input representation expected by a backend business application.

The EDI standard is monolithic, complicated, and coordinated through a heavy committee process. As a result, high costs are introduced in the way of expensive translation software, consultant and dedicated staff required for implementation and maintenance. Even with these disadvantages, EDI is still currently used in SCM for sharing information and communication.

The XML acronym stands for eXtensible Markup Language. XML can represent almost any type of information. It is relatively easy to learn, and many inexpensive tools are available for working with XML-formatted data. In response to these benefits, XML-based technologies are infusing major emerging e-business initiatives and standards, ranging from B2B documents, application interfaces, B2B communication protocol and complete framework.

As we know, the transactions in the supply chain need to have real-time response.

However, both EDI and XML documents are often processed in batch. This causes the information of each entity in the supply chain is not refreshed timely and up-to-date. In other words, supply chain business processes by using EDI and XML do not really support real time response.

Web Service is a new technology for companies to integrate the business software applications within the organization and with trading partners. Web Services use standard protocols and data formats such as HTTP, SOAP, and XML to connect to other software applications. This enables the information in the supply chain can be exchanged easily and in real-time manner.

## **2.4 Web Services**

Web Services are the consequence of the evolution of the Web that provides the means for software to connect to other software applications through standard protocols and data formats over the internet. Web services are concerned with the problems of enabling systematic application-to-application interactions over the Web, and are focused on interoperability, support for efficient application integration, and the creation of a uniform representation of applications within heterogeneous distributed systems.

In this section, Web Services will be defined and discussed. The technologies that make up the Web Services platform will also be described.

### **2.4.1 Introduction**

Web Services model is built upon existing industry standards such as XML. By using XML-based messaging as the mechanism, both the Web Services client and the Web Services provider are freed from needing any knowledge of each other beyond inputs, outputs and location. By using ubiquitous and standard internet protocol, Web Services can easily provide interoperable software functions over the internet and the intranet. Since it is accessible through a standard interface, Web Services allow



heterogeneous systems to work together as a single component. Web Services are enabling a new era of distributed application development.

#### **2.4.2 Definition of Web Service**

Web Service was defined by the World Wide Web Consortium (W3C) Web Services Architecture Working Group (Hugo, 2004) as:

“A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. “

(IBM Web Services Architecture Team, 2000) defined Web Service as:

“A Web Service is a collection of functions that are packaged as a single entity and published to the network for use by other programs. Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the World Wide Web.

According to Microsoft Developer Network (MSDN) (Roger, 2001), there are probably as many definitions of XML Web Service as there are companies building them, but almost all definitions have these things in common:

- XML Web Services expose useful functionality to Web users through a standard Web protocol. In most cases, the protocol used is SOAP.
- XML Web services provide a way to describe their interfaces in enough detail to allow a user to build a client application to talk to them. This description is usually provided in an XML document called a Web Services Description Language (WSDL) document.
- XML Web services are registered so that potential users can find them easily. This is done with Universal Discovery Description and Integration (UDDI).

Web Service was defined by Sun Microsystems as (Anne, 2001):

“A Web service represents a unit of business, application, or system functionality that can be accessed over the Web. Web services are applicable to any type of Web environment, be it Internet, intranet, or extranet, with a focus on business-to-consumer, business-to-business, department-to-department, or peer-to-peer communication.”

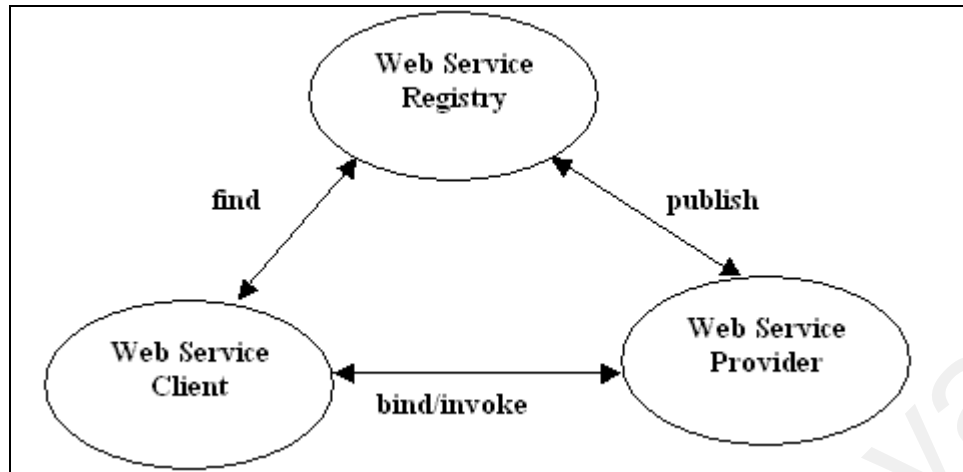
Webopedia Computer Dictionary gave the definition of Web Service as (Computer Dictionary , 2005):

“The term Web Services describes a standardized way of integrating Web-based application using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available.”

Based on the above definitions, Web Service can be defined as “a standardized way to support interoperable interaction over a network using the XML, SOAP, WSDL and UDDI open standards”.

### **2.4.3 Web Service Technologies**

The Web Services model follows the publish, find, and bind paradigm. Web Service providers publish services to a Web Service client. Web Service clients find required services using a Web Service Registry and bind to them. These ideas are shown in the figure 2.1 (IBM Web Services Architecture Team, 2000).



*Figure 2.1: The publish, find, and bind paradigm of Web Services (IBM Web Services Architecture Team, 2000)*

The Web Services platform needs a set of features to enable building distributed applications. To enable interoperability, a standard data representation format and data type system must be provided by the Web Services platform. This allows the Web Services to communicate among different type of platforms, programming languages and component models. In this context, XML and XML Schema (XSD) are used to represent data information and specify message formats respectively.

Traditionally, interface-based platforms for distributed systems have to describe the interfaces, methods, parameters and return value in order for others to invoke. Web Services platform uses Web Services Description Language (WSDL) as a means for describing Web Services and for providing the information others need to call.

Besides, there must be a mechanism to invoke the Web Services remotely as like Remote Procedure Call (RPC) Protocol. The Simple Object Access Protocol (SOAP) is the channel used for communication between a Web Services provider application and a client application.

Finally, in order for applications to quickly and easily find and bind Web Services over the Internet, a standard interoperable platform must exist. The Universal

Description, Discovery, and Integration (UDDI) is the standard used to solve the problems of registering, finding and binding of the Web Services.

The following sub sections describe in details the technologies, namely XML, WSDL, SOAP and UDDI, used in making up the Web Services platform.

#### **2.4.3.1. eXtended Markup Language (XML) and XML Schema**

Extended Markup Language (XML) is an extensible, portable, and structured text format. XML has a well-defined syntax and semantics that make it a simple and powerful, mechanism for capturing and exchanging data between different applications. It can represent almost any type of information. In response to these benefits, XML-based technologies are infusing major e-business initiatives and standards, such as B2B documents, B2B communication protocols and Web Services.

Web Services depend heavily on XML to communicate with each other even if they are using different information systems (MSDN, 2005). XML makes the data portable. XML is used in the Web Services architecture as the format for transferring information and data between a Web Services provider application and a Web Services client application.

The design goals for XML are (W3C Recommendation, 2004):

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.

8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

These design goals show that XML serves as a general framework for developing almost any special purpose business language. Parties, such as the e-business community, that share a common data-exchange problem can use XML as an open solution to that problem.

Schemas are used to describe the grammar and constraints on the elements of an XML document. Schemas provide a means of defining the structure, content, and semantics of XML documents that can be used to identify the document rules and validate the XML documents.

One of the common forms of schema for defining the rules for XML documents is the Document Type Declaration (DTD). DTDs are like templates that express the rules and constraints of an XML, such as name and order of the elements, attributes and type of values that an element or attribute may contain. XML documents are the actual instances of the template constructed following the rules. Each particular XML document created can be tested against the rules explicitly defined by the DTD.

Although a DTD can define the structure of the XML document, it cannot perform robust data-type checking of content. In response to this issue, XML Schema has been developed. XML Schema specification improves greatly upon the DTD content model by providing rich datatyping capabilities for elements and attributes as well as providing Object-Oriented design principles (Galloway, 2002).

The two major goals that the W3C XML Schema working group focused on during the design of the XML Schema standard were (W3C Recommendation, 2004):

- Expressing Object Oriented design principles found in common Object-Oriented programming languages into the specification.

- Providing rich datatyping support similar to the datatyping functionality available in most relational database systems.

The XML Schema specification allows for the use of both built-in and custom defined data types making it possible to more accurately express and constrain data found in compliant XML documents. This provides a flexible and useful mechanism for automatic data validation.

#### **2.4.3.2. Simple Object Access Protocol (SOAP)**

Simple Object Access Protocol (SOAP) is a lightweight XML-based protocol for exchanging structured information in a decentralized and distributed environment (Martin, 2003). SOAP was developed for compatibility with many different platforms and operating systems. It enables machine-to-machine communication in heterogeneous environments. SOAP defines the minimal syntax and semantics of XML messages that are used for service invocation, response and fault reporting.

There are several different types of messaging patterns in SOAP. One of the most common is the Remote Procedure Call (RPC) messaging pattern. By using RPC, one network client node sends a request message to the server node, and the server immediately replies with a response message to the client. SOAP does not define any new transport protocol. Instead, it reuses the Hyper Text Transfer Protocol (HTTP) for transporting data as messages. Web Services use the SOAP as the channel for communication between the provider application and the client application. The use of HTTP as the underlying protocol ensures that Web Services provider applications and client applications can communicate using the Internet.

#### **2.4.3.3. Web Services Description Language (WSDL)**

WSDL stands for Web Services Description Language. According to W3C, WSDL is “an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The

operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate.”

WSDL is an XML-based language for describing operational features of Web Services. WSDL descriptions are composed of interface and implementation definitions. The interface is an abstract and reusable service definition that can be referenced by multiple implementations. The separation of interface from implementation allows the implementation to change while the interface remains the same.

WSDL represents a contract between the service requestor and the service provider. The crucial difference is that WSDL is platform- and language-independent and is used primarily to describe SOAP services. Basically, WSDL describes four critical pieces of data (Ethan, 2002):

- Interface information describing all publicly available functions
- Data type information for all message requests and message responses
- Binding information about the transport protocol to be used
- Address information for locating the specified service

By using WSDL, a service requestor can locate a Web Service and invoke any of its publicly available functions. WSDL therefore represents a cornerstone of the web service architecture because it provides a common language for describing services and a platform for automatically integrating those services (Ethan, 2002).

#### **2.4.3.4. Universal Description, Discovery, and Integration (UDDI)**

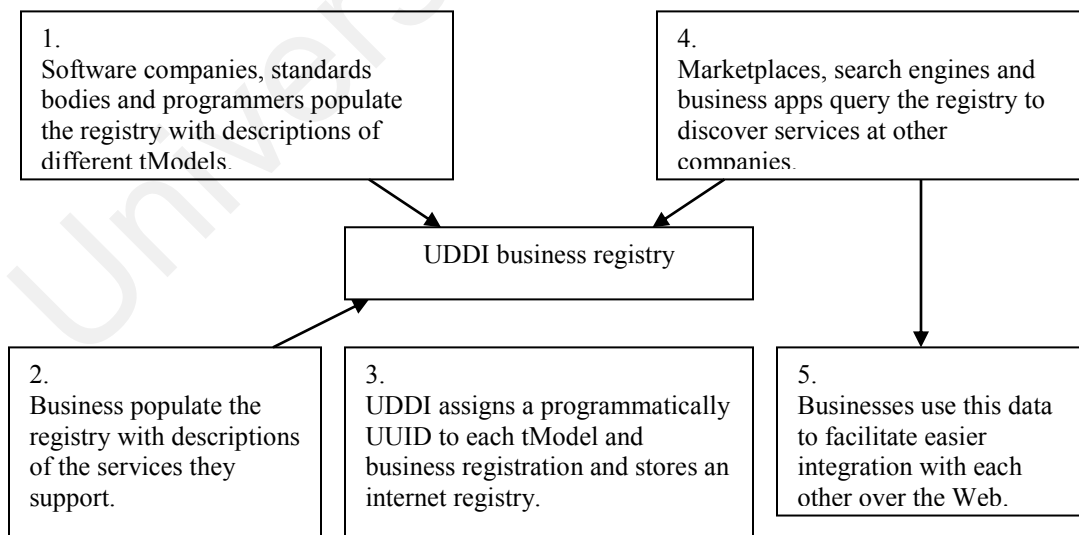
The Universal Description, Discovery, and Integration (UDDI) protocol is a key member of the group of interrelated standards that comprise the Web Services stack. It defines a standard method for publishing and discovering the network-based software

components of a service-oriented architecture (SOA) (UDDI.ORG, 2004).

UDDI is a specification for creating distributed Web-based registries of Web services. A UDDI registry stores information on businesses, the services offered by these businesses, and technical information about these services. The data model and application programming interface(API) used by UDDI, based on XML and SOAP, enable companies and applications to quickly, easily and dynamically find and use Web Services over the Internet.

A UDDI registry can be compared to an Internet search engine which contains indexed and categorized information about pages available on the World Wide Web. However, where an Internet search engine returns only a URL to a Web page, a UDDI registry needs to return not only the location of the services, but also information about the service, how it functions, which parameters to use, its return values and so on.

UDDI provides a programming model and schema, which define the rules for communicating with the registry. All Application Programming Interfaces (APIs) in the UDDI specification are defined in XML, wrapped in a SOAP envelope and sent over HTTP.



*Figure 2.2: How UDDI works (Tom, 2002)*



Figure 2.2 illustrates how a UDDI registry is populated with data and how customers discover and use this information. A UDDI registry is built on the data provided by its customers. There are several steps to making data useful in UDDI. As shown in step 1, publishing useful information to the registry begins when software companies and standards bodies define specifications relevant to an industry or business, which they register in UDDI. These are known as technical models, or more commonly as tModels.

In step 2, companies also register descriptions of their businesses and the services they offer. A UDDI registry keeps track of all these entities by assigning each one a programmatically unique identifier, known as a Unique Universal Identifier (UUID) key as shown in step 3. A UUID key is guaranteed to be unique and never changes within a UDDI registry.

Other clients, such as e-Marketplaces, search engines, and business applications in step 4, use a UDDI registry to discover services of interest. In turn, other businesses may invoke these services, allowing simple and dynamic integration as shown in step 5.

#### **2.4.3.5. Microsoft .NET Framework**

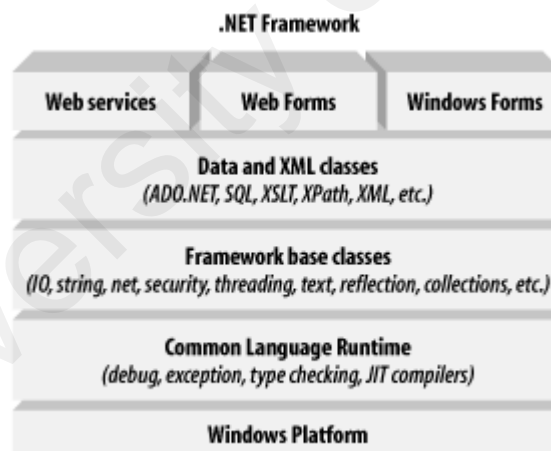
The .NET Framework is a general-purpose software development platform that allows different programming languages and libraries to work together seamlessly to create applications. It provides development tools, run-time environments, server infrastructure, and a consistent object-oriented programming model for creating software that are easier to build, manage, deploy and integrate with other systems. In addition, .NET provides remoting infrastructure that allows applications running in different processes on the same or different computers to exchange data using binary or HTTP protocols (Microsoft .NET, 2005).

The .NET Framework is designed to fulfill the following objectives (MSDN, 2005):

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.

- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

Figure 2.3 breaks down the .NET Framework into its architectural components. The .NET Framework consists of two main parts, i.e.: Common Language Runtime (CLR) and the Base Class Library.



*Figure 2.3: The .NET Framework architectural components (MSDN, 2005)*

- **The Common Language Runtime (CLR)**

The CLR is the foundation of the .NET Framework. It is an agent that responsible for run-time services such as language integration, security enforcement, and memory, process, and thread management and other forms of code accuracy that promote security

and robustness (MSDN, 2005). In addition, features such as life-cycle management, strong type naming, cross-language exception handling, and dynamic binding in CLR helps in reducing the amount of code that a developer need to write to turn business logic into a reusable component.

The CLR is designed to enhance performance. Although the CLR provides many standard runtime services, the managed code is never interpreted. Code that targets the runtime is known as managed code. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing. The CLR is responsible for providing the execution environment that code written in a .NET language runs under.

The .NET Framework supports Common Language Specification (CLS) that provides the necessary foundation for language interoperability (Liberty, 2003). For example, classes written in C# and can be derived in VB.NET. The CLR provides all managed code with support for executing in a multi-language environment. The choice of language would be a personal preference rather than a limiting factor in the application development.

- **Base Class Library**

The base classes provide standard functionality such as input output, string manipulation, security management, network communications, thread management, text management, and user interface design features. The base class library is a comprehensive and object-oriented collection of reusable types that tightly integrate with the common language runtime. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. (MSDN, 2005)

The class library includes types that support a variety of specialized development scenarios. The ADO.NET classes enable developers to interact with data accessed in the form of XML through the OLE DB, ODBC, Oracle, and SQL Server interfaces. XML

classes enable XML manipulation, searching, and translations. The ASP.NET classes support the development of Web-based applications and Web services. The Windows Forms classes support the development of desktop-based smart client applications. Together with all the class libraries, .NET framework provides a common and consistent development interface across all languages supported.

## **2.5 Existing Projects**

### **2.5.1 Electronic Business using eXtensible Markup Language (ebXML)**

Electronic Business using eXtensible Markup Language (ebXML) is a modular suite of specifications sponsored by Organization for Advancement of Structured Information Standards (OASIS) and United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) that enables enterprises of any size and in any geographical location to conduct business over the Internet (OASIS, 2005). The specifications cover the analysis of business processes and business documents, the documentation of a company's capabilities and the document transfer to conduct E-Business. Compared to Web Services, ebXML is more at the executive business level (Alonso, *et. al.*, 2003). ebXML provides an entire architecture that defines a new way of thinking about business and documenting the business itself.

With ebXML, companies are able to define how to conduct business using a specific vocabulary. Core components are used to build predefined documents. Messages are sent using standardizes protocols and formats. All of this information is stored in ebXML registries. Business Processes and Business Document has to be created prior to their use. A Business Process Specification describes how a business works and a Business Document Specification describes the information exchanged between the partners. Both of these documents are defined by using the Business Process Specification Schema (BPSS). To facilitate their creation and avoid reinventing the wheel, these documents can be composed of reusable and extendable Core Components. Besides, ebXML Messaging Service (ebMS) is used to provide a simple way to exchange

business documents using standard protocols. It is based on the SOAP specification, but provides additional headers and envelopes to include information about transport, routing, policies and security. Moreover, an ebXML Registry provides means for finding organizations, business processes, core components and other objects. Therefore it does not store the actual objects but metadata about and associations between them (Gerstbach, 2005).

By using ebXML, companies are able to define how to conduct trading relationships and communicate data in common terms. ebXML provides a methodology for business to determine what information they should exchange and how, as well as a set of specifications to allow automation of the process. Figure 2.3 shows an ebXML system overview that involves the following steps (Chase, 2002):

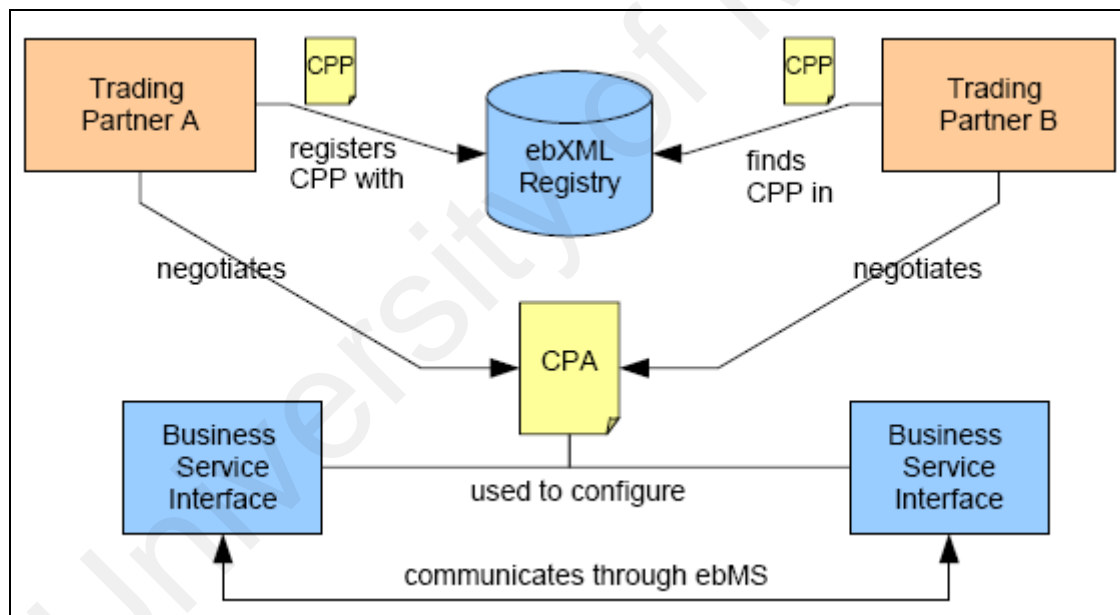


Figure 2.4: ebXML System Overview (Chase, 2002)

1. **Search for a Trading Partner.** An ebXML registry contains information on potential trading partners. During the search the registry is queried for a trading partner that offers the wanted service. Services are described using Collaboration Protocol Profiles (CPPs).

2. **Create an Agreement.** Based on the CPPs of both partners a Collaboration Protocol Agreement (CPA) is composed. It specifies what kind of business is to be performed and how. Therefore it includes information on technical issues such as protocols and requirements regarding security or acknowledgments. Usually the CPA is negotiated after being proposed by one party.
3. **Configure both Business System Interfaces using the CPA.** Based on the agreement it is now possible to configure an ebXML enabled application.
4. **Begin performing Business Processes.** The final step is the execution of the process. Based on the fact that both partners use the same documents describing the business process (the CPA) collaboration between them is possible.

From the point of view of implementation, ebXML is a much more complex system than Web Services. Indeed, it is true that Web Services-enabled systems can be implemented very quickly if developers use the existing, powerful libraries, such as Microsoft .NET libraries. Those libraries allow for developers to accomplish technology-centric integration assignments quickly. While it is possible to have a simple Web Service up and running within minutes, a simple ebXML system will require much more efforts and time. Accordingly, small technology-oriented integration scenarios remain the strength of Web Services.

ebXML is a domain-specific vertical framework while Web Services are a general horizontal framework for Service Oriented Architecture (SOA). The specification of ebXML is not entirely complete and the industry support of ebXML is still lacking. If industry fails to provide affordable implementations of ebXML, this standard might not be widely adopted due to its cost and industry support.

### 2.5.2 Web Service Intelligent Agent Structuring for Supply Chain Management

An intelligent agent architecture based on Web Services has been proposed by Umair and Ben (Umair and Ben, 2005). These agents utilize XML as communication mechanism and are deployed on each organization. The basis of the proposed agent model is to share common knowledge among supply chain partners and to update and adapt to the business environment. These agents provide a mechanism that is used to store and share the centralized and distributed data across the supply chain. Figure 2.5 shows how the agent is structured. The bottom most part is the UDDI Registry that contains information about all the agents in the organization and what they are capable of. This information is necessary for other agents to communicate with the targeted organization through the SOAP protocol. The WSDL component defines the functions or operations that are supported by the agents. The information that is required for agent to communicate with each other is in the WSDL. It is up to the contacting agent to interpret the information for defining the template of SOAP message (Umair and Ben, 2005).

Database	Problem Solving Methodologies	Courses of Actions	Business Constraints Controller
Web Services Agents			
WSDL			
UDDI Registry			

*Figure 2.5: Proposed Web Service Agent (Umair and Ben, 2005)*

Once the agent service is instantiated it can perform the assigned tasks by using

the other four components in the model (Umair and Ben, 2005):

- **Database**

It contains information about previous experiences of collaboration with organization, parent organization's information like inventory levels, order information, logistic information and etc.

- **Problem Solving Methodologies**

This component contains information about the methods available to overcome a problem. For instance, if a supplier is not capable of supplying the required quantity, the order should be void.

- **Courses of Actions**

This component contains the available courses of actions once a function is to be performed. For example, if an order is received the receiving, agent should inform the manufacturing unit and obtain information about the progress in a stepwise manner.

- **Business Constraints Controller**

Business constraints are defined by an organization and vary, depending on an organization's state.

### **2.5.3 Web Service-based Framework for Supply Chain Management**

A Web Service-based Supply Chain Management framework which has been proposed by Zhang Mi (Zhang Mi et al., 2005) is shown in Figure 2.6. The framework is designed based on Web Services, agents and XML technologies. The key module in this architecture is the intermediate module that consists of a suite of agents. This module deals with UDDI organization and management, as well as different processes of supply chain transactions.

There are three levels in this framework. The first level is the intermediate module that processes the transactions. The organizations complete their business transactions



through this module. The second level consists of the users, i.e., business entities in the supply chain including Product Producer, Manufacturer, Provider, Distribution center, Sale center, Retailers and consumers. Each entity is comprised of many companies or individuals who have some business conducts through the supply chain. The third level is the lowest level, namely the basic databases supplied for those Web services to obtain the real data (Zhang Mi et al., 2005).

When a request is submitted, it is transferred to an agent first to read the UDDI organized structure and find an appropriate UDDI discovery order. The paper suggests a Peer-To-Peer (P2P) search model to search for the right UDDI and find the wanted information. After that, the request is forwarded to the targeted supply chain entities that provide the real Web Services for requesters. Finally the requester builds a real connection to the selected Web service provider. The Web service communicates with the background database systems to obtain the required data.

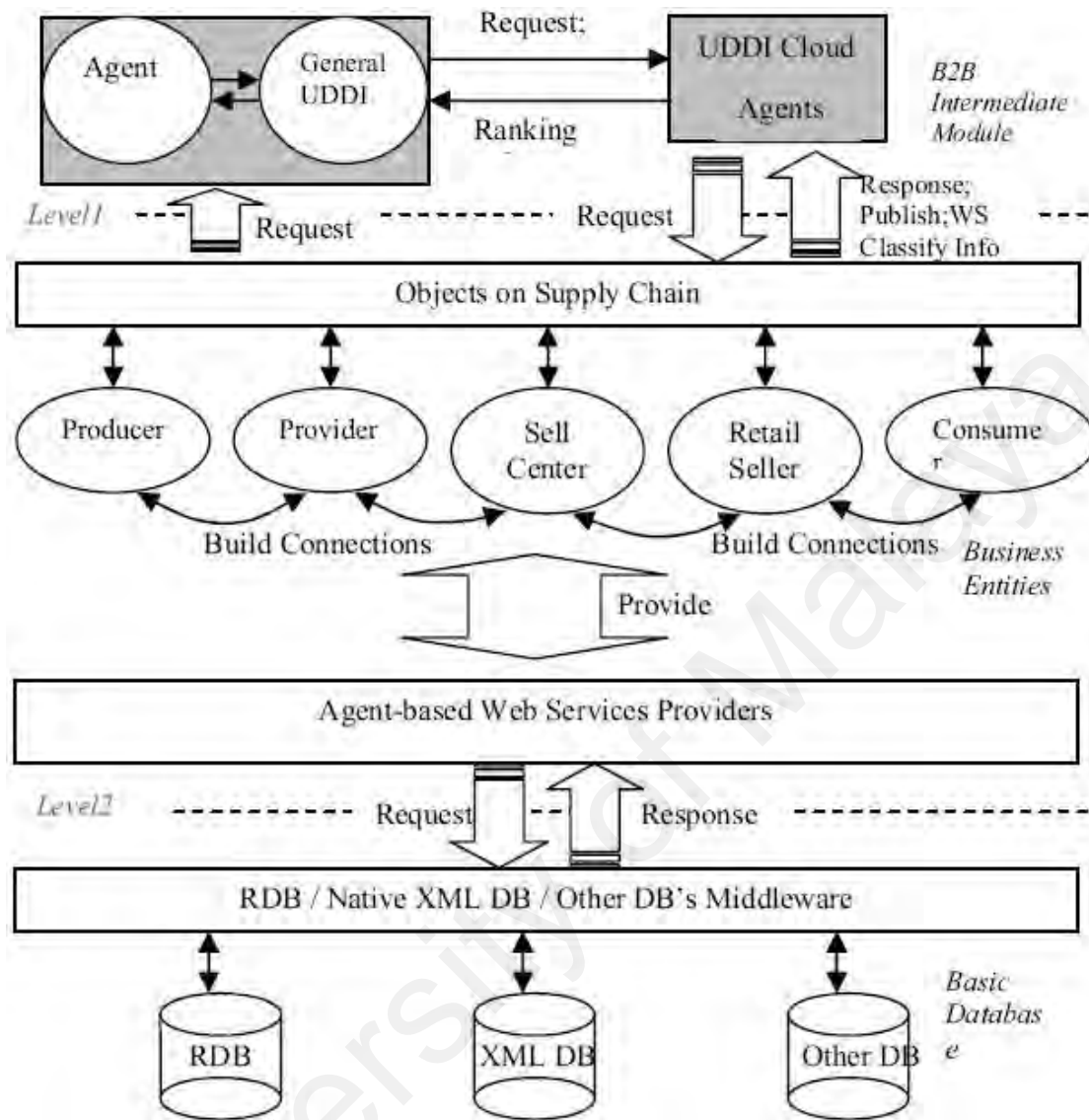


Figure 2.6: Framework of Supply Chain Management proposed (Zhang Mi et al., 2005)

#### **2.5.4 Comparison of the Proposed Framework With Others (ebXML, Umair and Ben, Zhang Mi et al.)**

Web Services for E-Procurement (WSEP) System is implemented using Web Service which is a well-adopted standard for system integration throughout most of the business sectors. Compared to ebXML, it can apply to most of the industry sector. Besides, implementation of the WSEP is not as complicated as ebXML. For many integration projects, organizations do not need full grown e-Business suites. Instead, they need smaller, more reliable, and easier to handle technologies, such as Web Services, that have reached a sufficient level of maturity. (Yan and Klein, 2005)

All of these frameworks are using registry or UDDI to register Web Services. UDDI design center is a high level business or service registry. It is focused on the discovery aspect of businesses, Web Services, and the technical interfaces they make available. The support for and use of UDDI is currently very limited (Clabby, 2002). If the needs are more than simple publish and discovery of business or service metadata, UDDI should not be used.

From the perspective of E-Procurement, the public UDDI is not suitable for the proposed WSEP System since it does not provide sufficient information, such as product information. Therefore a private database system has been used, as compared to UDDI, to store product and supplier information as well as supplier Web Services URI. A database, in this case, is easier to maintain and is more efficient.

## **2.6 Conclusion**

In this chapter, Supply Chain, SCM and E-Procurement are defined. Besides, the purchasing processes in the context of Supply Chain Management are reviewed and discussed. This chapter also defines and reviews the Web Services technologies and Microsoft .NET framework. In addition, three of the existing projects that are related to E-Business are reviewed and are compared with the WSEP System.

## **CHAPTER 3: METHODOLOGY**

Methodology can be defined as the branch of philosophy that analyzes the principles and procedures of inquiry in a particular discipline (WordNet, 2005). In software engineering, methodology is a framework of principles, techniques and procedures that defines different phases of the development process, such as planning, requirements analysis, design, testing and maintenance.

### **3.1 Research Methodology**

The research methodology that was used for this dissertation consists of 4 phases, as shown in Figure 3.1. Phase I is knowledge and requirements gathering which consists of literature review and research findings. Other methods such as historical analysis, interviews, surveys and questionnaires are not suitable for this dissertation. It is due to the end users or purchase officers are not exposed in the technologies behind such as Web Services that are used to integrate the Supply Chain. No numerical and statistical data needs to be collected and analyzed. Therefore, literature and document review was the only methodology that was used extensively for this dissertation.

Phase II of the research methodology consists of the design of the main components and system architecture based on the related work. This includes designing the flow of logic within the system. Phase III is the development and integration of the components to ascertain that the architecture that uses Web Services can provide dynamic procurement operations. The final phase is unit testing and documentation. If the testing result is unsatisfied, the system needs to be redesigned and develop.

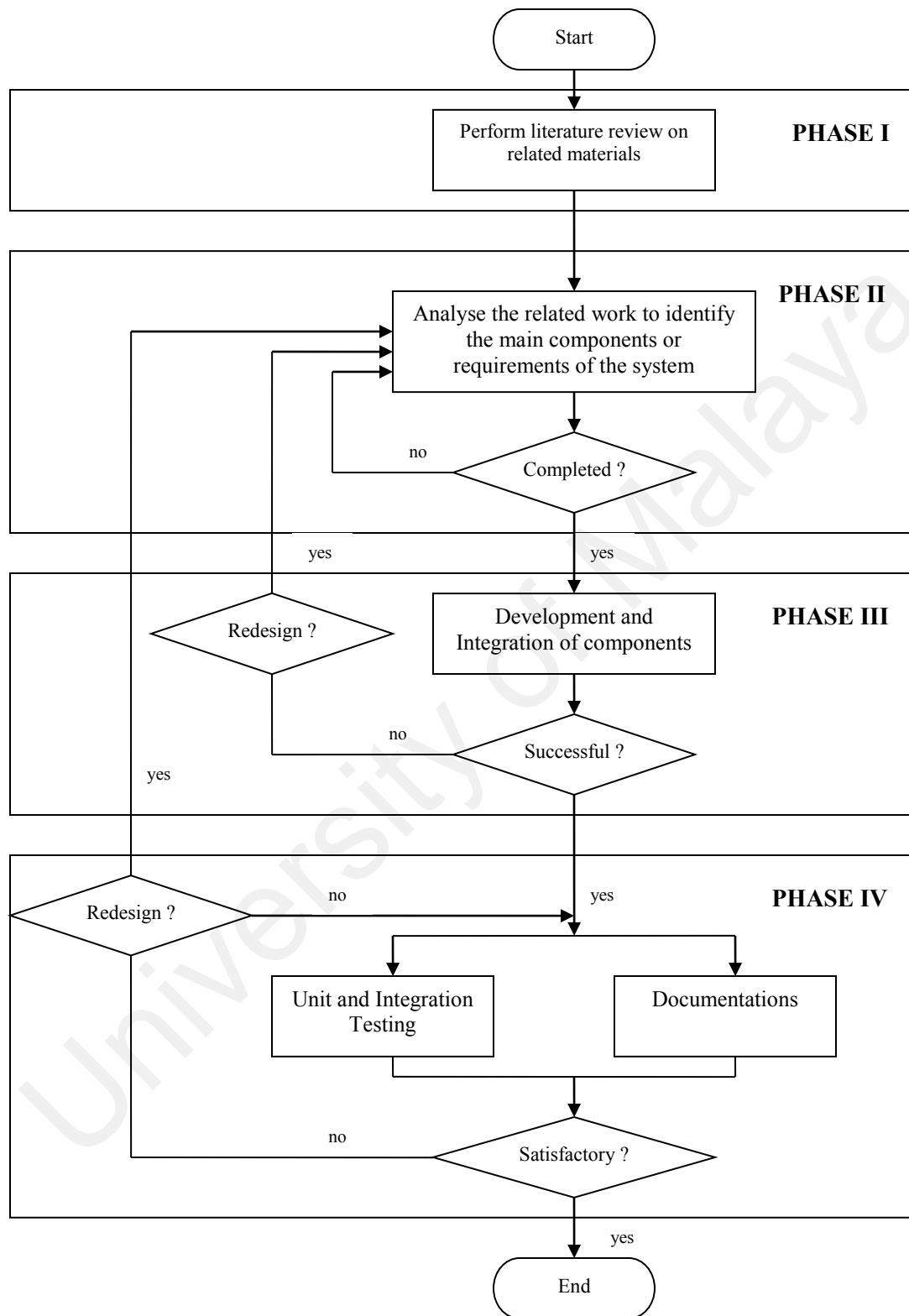


Figure 3.1: Research Methodology

### **3.2 Development Methodology**

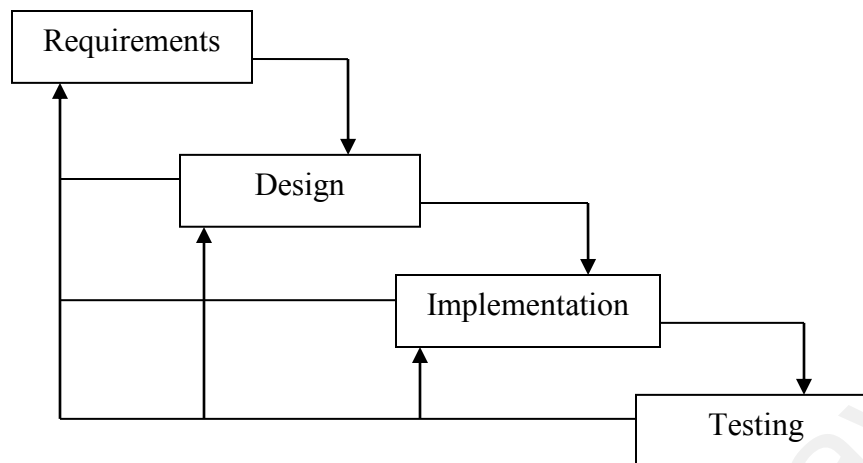
The software process model can be defined as a networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution (Marciniak, 2001). Each stage of the software process is identified and a model is then employed to represent the inherent activities associated within that stage. There are a few software process models that serve as an abstract representation of the software process. These include:

- waterfall model
- spiral model
- prototyping model
- extreme programming

#### **3.2.1 Choice of Modified Waterfall Model**

By looking at the above software process models, this project has taken the Waterfall Model coupled with iterative development, as shown in figure 3.1. These modifications tend to focus on allowing some of the stages to overlap, thus reducing the documentation requirements and the cost of returning to earlier stages to revise them. The Modified Waterfall Model uses the same phases as the pure waterfall, but is not done on a discontinuous basis. This enables the phases to overlap and provide feedback between phases. Overlapping stages, such as the requirements stage and the design stage, make it possible to integrate feedback from the design phase into the requirements.

With iterative development, the project is divided into small parts. Each iteration is actually a mini-Waterfall process with the feedback from one phase providing vital information for the design of the next phase. The software products which are produced at the end of the series of steps can go into production immediately as incremental releases.



*Figure 3.2: Modified Waterfall Model with iterative development.*

#### **3.2.1.1 Phase 1 – Requirements**

The first phase of the development is to gather and identify the requirements of the system. The problem is specified along with the desired objectives. The requirements specification is then produced from the detailed definitions that clearly define the system functions. The full details of the specification are described in the next chapter, which includes functional requirements, non-functional requirements and hardware requirements.

#### **3.2.1.2 Phase 2 – Design**

The next phase of the development is system design. System design is a phase that emphasizes on how the system should perform in order to fulfill the requirements identified in the analysis phase. This involves architecture design, objects and classes design, database design and user interface design. Unified Modeling Language (UML) modeling, such as Sequence Diagrams along with Class Diagrams have been used in this project to specify, model and document the system.

A new database needs to be designed and implemented in later phase. Database

schema or data dictionary needs to be created during this phase. The details of the database design are covered in Chapter 5.

### **3.2.1.3 Phase 3 – Implementation**

The implementation phase involves the actual development of the system. With the documentations from the analysis and design phase, the system should be built upon what has been documented.

The implementation phase deals with issues of quality, components and debugging. The development involves the implementation of User Interface, Business Logic Tier, Data Access Tier and the integration of all these components. The end deliverable of this phase is the WSEP System. Chapter 6 describes the implementation phase in details.

### **3.2.1.4 Phase 4 – Testing**

The final stage in this process model is testing. The system needs to be tested for conformance with the system requirements. The first stage in the testing process involve unit testing. The second stage will be module testing which involve multiple units to be tested together. When all these are done, the integration testing phase begins, combining all the individual modules together. Finally the whole system is tested.

## **3.3 Techniques Used to Define Requirements**

Various techniques have been used to define the requirements of the system. These techniques are important in gathering various requirements of the system that will be essential building blocks for subsequent developments. These techniques are as follow:

- Analysis of existing projects



- Review of new technologies
- Library Research and Internet Research

### **3.2.1 Analysis of existing projects**

Research is done based on internet journal and literature review. Journals or white papers published with similar systems are studied and reviewed. This helped to compare and evaluate their system with the proposed architecture.

### **3.2.2 Review of new technologies**

Reviewing new technologies like Web Services and .NET frameworks will give further understanding of the new features and capabilities. This helps in determining of requirements that could be included into the system based on the technologies supported.

### **3.2.3 Library Research and Internet Research**

Library and Internet provide lots of information that is useful for the research of the project. Books could either be browsed or borrowed from the library to gather information that would be useful for the development of this project. The library in the University of Malaya provides valuable of resources to develop this project. Digital library, such as IEEE journal and ACM, helps in finding useful information for writing this report.

There are many tutorials and articles that are available for download from the Internet. White papers from various vendors are valuable information that often can be downloaded for free. The use of search engines such as Google has proven to result in vital information which could not be found in books.

## CHAPTER 4: SYSTEM REQUIREMENT ANALYSIS

System Requirement Analysis is the detailed documentation of the system services and constraints. The goal of the requirements phase of software development is to decide precisely what to build and document the results. This section lays out important concepts and discusses capturing functional and non-functional requirements so that they can drive the architectural decisions and be used to validate the architecture.

### 4.1 Functional Requirements

Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. The WSEP system can be divided into four main modules:

- **Authentication Module**

Authentication module is responsible for user authentication and authorization of the system. This involves accepting username and password, and validating them against database. The module enables single sign-on for users to access the WSEP system. If an unauthenticated user tries to access a page, the Authentication Module will redirect the user to the login page to request for credential.

- **Maintenance Module**

One of the functions of the system is to manage the system's entities which include products, suppliers and suppliers' Web Services Uniform Resource Identifier (URI). The Maintenance Module should allow the user to perform CRUD (Create, Read, Update, Delete) operations against the entities in the database.

- **E-Procurement Transaction Web Services Module**

This module is the main component that provides real-time information sharing and dynamic procurement operations. It interacts with the suppliers' Web

Services to request for quotations as well as submit purchase orders. The operations should be transparent where the users do not need to know what the supplier business systems are all about.

- **Suppliers' Web Services Modules**

The Suppliers' Web Services are necessary as part of the WSEP System. All the suppliers who want to participate in the E-Procurement System have to implement Web Services that conform to the published Web Services Description Language (WSDL) format. The suppliers are not restricted to any Web Services platform for implementing this module since Web Services can easily communicate with each other no matter what language is used.

In this project, use case diagrams have been used for capturing the functional requirements. A use case defines a goal-oriented set of interactions between external actors and the system under consideration. The following sub sections describe in details all the use cases in the WSEP system.

#### **4.1.1 Use Case Diagram**

Use case diagrams model the functionality of system using actors and use cases. Figure 4.1 shows all the use cases that capture the intended behavior of the system. The actor, Purchase Officer, is required to authenticate himself in order to use the system. The Purchase Officer is responsible to maintain supplier, maintain product, maintain suppliers' Web Services URI, request for quotation and raise purchase order. For Maintain Supplier, Maintain Product and Maintain Supplier WS, each of which is specialized with List, Add, Edit and Delete use cases. Request For Quotation and Raise Purchase Order use cases have an extending use case called Query Supplier Web Services and Invoke Supplier Web Services respectively. The details behind each element on the use case diagram are captured in textual form as the following.

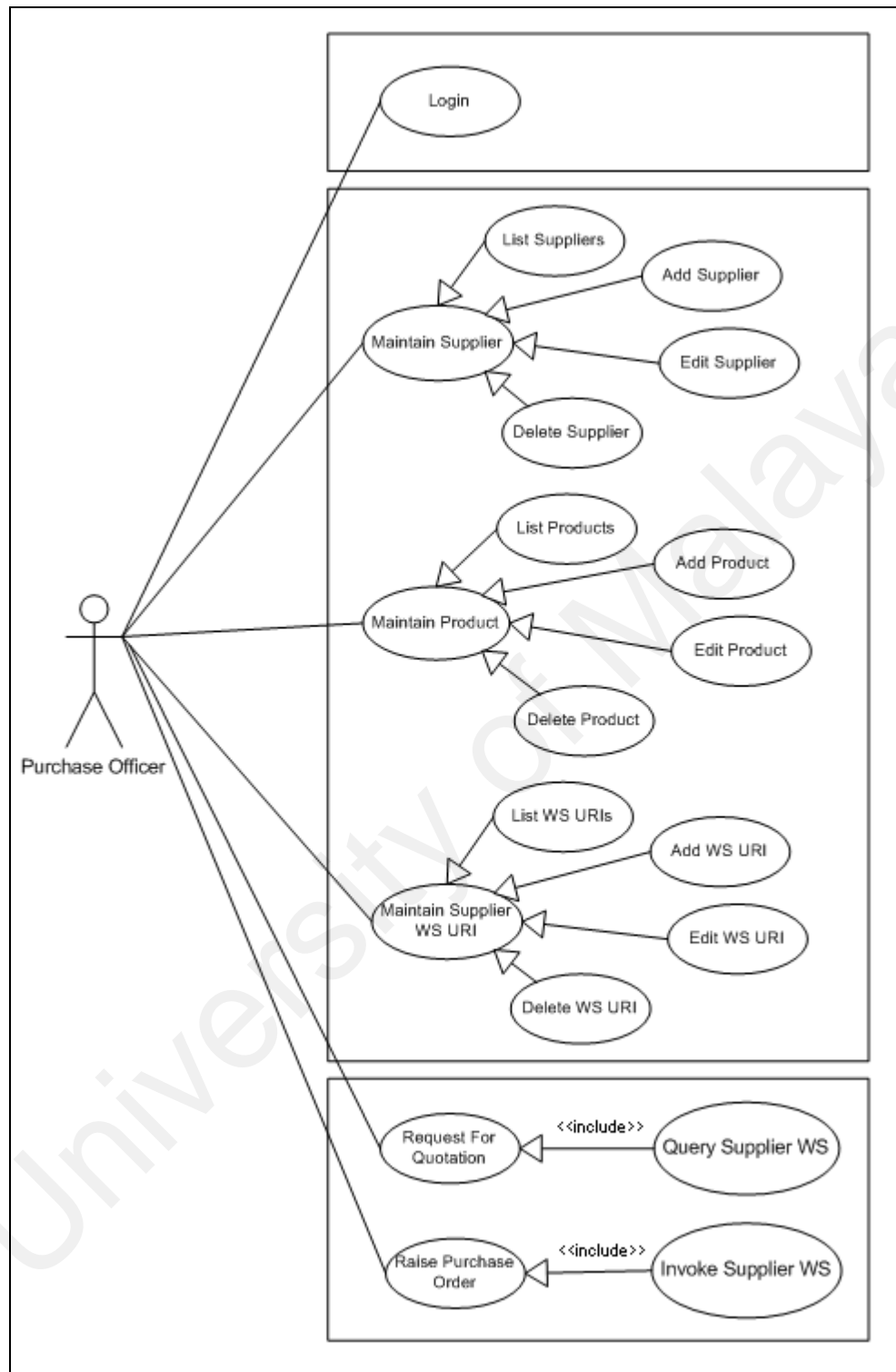


Figure 4.1 Use Case Diagram of WSEP System

### Use Case 1: Login

<b>Goal of Use Case:</b>	System authenticates the User	
<b>Preconditions:</b>	Username and password are provided.	
<b>Success Post Conditions:</b>	The Purchase Officer is authenticated to use the system.	
<b>Failed Post Conditions:</b>	The Purchase Officer cannot enter into system.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	Enter username and password.
	2	Submit the credential to the system for authentication.

### Use Case 2: List Suppliers

<b>Goal of Use Case:</b>	The Purchase Officer lists all the suppliers.	
<b>Preconditions:</b>	None.	
<b>Success Post Conditions:</b>	All the suppliers are listed.	
<b>Failed Post Conditions:</b>	No supplier information is returned.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	List all the suppliers

### Use Case 3: Add Supplier

<b>Goal of Use Case:</b>	The Purchase Officer adds a supplier.	
<b>Preconditions:</b>	Supplier details are known.	
<b>Success Post Conditions:</b>	Supplier details are added successfully into database.	
<b>Failed Post Conditions:</b>	Supplier details are not added into database.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	Purchase Officer enters the supplier information
	2.	Purchase Officer submits the information and save to database

### Use Case 4: Edit Supplier

<b>Goal of Use Case:</b>	The Purchase Officer edits supplier information.	
<b>Preconditions:</b>	Supplier details are known.	
<b>Success Post Conditions:</b>	Supplier details are successfully saved to database	
<b>Failed Post Conditions:</b>	Supplier details are not saved to database.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	Purchase Officer modifies supplier information
	2.	Purchase Officer submits the information and save to database

### Use Case 5: Delete Supplier

<b>Goal of Use Case:</b>	The Purchase Officer deletes a supplier	
<b>Preconditions:</b>	Supplier to be deleted is identified	
<b>Success Post Conditions:</b>	Supplier is successfully deleted from database	
<b>Failed Post Conditions:</b>	Supplier is not deleted from database.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	Purchase Officer identifies the supplier to be deleted
	2.	Purchase Officer delete the supplier from database

### Use Case 6: List Products

<b>Goal of Use Case:</b>	The Purchase Officer lists all the products.	
<b>Preconditions:</b>	None.	
<b>Success Post Conditions:</b>	All the products are listed.	
<b>Failed Post Conditions:</b>	No product information is returned.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	List all the products

### Use Case 7: Add Product

<b>Goal of Use Case:</b>	The Purchase Officer adds a product.	
<b>Preconditions:</b>	Product details are known.	
<b>Success Post Conditions:</b>	Product details are added successfully into database.	
<b>Failed Post Conditions:</b>	Product details are not added into database.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	Purchase Officer enters the product information
	2.	Purchase Officer submits the information and save to database

### Use Case 8: Edit Product

<b>Goal of Use Case:</b>	The Purchase Officer edits product information.	
<b>Preconditions:</b>	Product details are known.	
<b>Success Post Conditions:</b>	Product details are successfully saved to database	
<b>Failed Post Conditions:</b>	Product details are not saved to database.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	Purchase Officer modifies product information
	2.	Purchase Officer submits the information and save to database



#### Use Case 9: Delete Product

<b>Goal of Use Case:</b>	The Purchase Officer deletes a product	
<b>Preconditions:</b>	Product to be deleted is identified	
<b>Success Post Conditions:</b>	Product is successfully deleted from database	
<b>Failed Post Conditions:</b>	Product is not deleted from database.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	Purchase Officer identifies the product to be deleted
	2.	Purchase Officer delete the product from database

#### Use Case 10: List Supplier Web Services URIs

<b>Goal of Use Case:</b>	The Purchase Officer lists all the supplier Web Services URI.	
<b>Preconditions:</b>	None.	
<b>Success Post Conditions:</b>	All the supplier Web Services URIs are listed.	
<b>Failed Post Conditions:</b>	No supplier Web Services URI information is returned.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	List all the supplier Web Services URIs

### Use Case 11: Add Supplier Web Services URI

<b>Goal of Use Case:</b>	The Purchase Officer adds a Supplier Web Services URI.	
<b>Preconditions:</b>	Supplier Web Services URI details are known.	
<b>Success Post Conditions:</b>	Supplier Web Services URI details are added successfully into database.	
<b>Failed Post Conditions:</b>	Supplier Web Services URI details are not added into database.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
<b>Description:</b>	<b>Step</b>	<b>Action</b>
	1.	Purchase Officer enters the Supplier Web Services URI information
	2.	Purchase Officer submits the information and save to database

### Use Case 12: Edit Supplier Web Services URI

<b>Goal of Use Case:</b>	The Purchase Officer edits supplier Web Services URI.	
<b>Preconditions:</b>	Supplier Web Services URI details are known.	
<b>Success Post Conditions:</b>	Supplier Web Services URI details are successfully saved to database	
<b>Failed Post Conditions:</b>	Supplier Web Services URI details are not saved to database.	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	

Description:	Step	Action
	1.	Purchase Officer modifies Supplier Web Services URI information
	2.	Purchase Officer submits the information and save to database

### Use Case 13: Delete Supplier Web Services URI

<b>Goal of Use Case:</b>	The Purchase Officer deletes a supplier Web Services URI.	
<b>Preconditions:</b>	Supplier Web Services URI to be deleted is identified	
<b>Success Post Conditions:</b>	Supplier Web Services URI is successfully deleted from database	
<b>Failed Post Conditions:</b>	Supplier Web Services URI is not deleted from database	
<b>Actors:</b>	Purchase Officer	
<b>Triggers:</b>	This process is started by the Purchase Officer (human interaction)	
Description:	Step	Action
	1.	Purchase Officer identifies the supplier Web Services URI to be deleted
	2.	Purchase Officer delete the supplier Web Services URI from database

#### Use Case 14: Request For Quotation

<b>Goal of Use Case:</b>	The Purchase Officer requests quotations from suppliers to replenish stock for a particular product.	
<b>Preconditions:</b>	<ol style="list-style-type: none"><li>1. The inventory level of a product has fallen below its minimum level</li><li>2. The Product ID is known</li></ol>	
<b>Success Post Conditions:</b>	The system returns a list of suppliers with the requested products pricing.	
<b>Failed Post Conditions:</b>	Null or error message is returned	
<b>Actors:</b>	Purchase Officer, Procurement Web Services	
<b>Triggers:</b>	Triggered external suppliers' Web Services to get products pricing.	
<b>Description</b>	<b>Step</b>	<b>Action</b>
	1	Purchase Officer is alerted that a product has fallen below its minimum level.
	2	Purchase Officer enters the Product Code and submits to the WSEP system.
	3	The system retrieves suppliers' WSDLs in the Database based on the Product ID.
	4	For each of the WSDLs retrieve, the system invokes suppliers' Web Services to get the product price, quantity on hand and other product information.
	5	The system returns a list of suppliers along with the requested products pricing.

### Use Case 15: Raise Purchase Order

<b>Goal of Use Case:</b>	The Purchase Officer place orders from selected suppliers to replenish stock for a particular product.	
<b>Preconditions:</b>	The suppliers are selected and the product quantities are entered.	
<b>Success Post Conditions:</b>	New Purchase Orders are created and are sent to the suppliers.	
<b>Failed Post Conditions:</b>	Null or error message is returned.	
<b>Actors:</b>	Purchase Officer, Procurement Web Service.	
<b>Triggers:</b>	None	
<b>Description</b>	<b>Step</b>	<b>Action</b>
	1	Purchase Officer selects the suppliers and enters the order quantity for each selected suppliers.
	2	Purchase Officer submits the Purchase Order to the selected suppliers.
	3	Purchase Orders are sent to the suppliers.

## 4.2 Non-Functional Requirements

Non-functional requirements define system properties and constraints. They specify criteria that can be used to judge the operation of a system rather than specific behaviors. They define the constraints under which the system must operate.

- **Dynamic and Real Time Requirements**

The system should provide real time product information retrieved directly from the supplier inventory systems. Any data updates in the supplier inventory systems should be reflected in the quotation to the purchase officer. Submission of purchase orders should happen dynamically based on purchase decision.

- **Security Requirements**

The system should ensure that data is protected from unauthorised access. This is done by providing user authentication and using secure channel like SSL to encrypt the data transmits. The system should provide a means to enter user names and passwords.

- **Performance Requirements**

The system should perform within a reasonably response time under defined circumstances. The only foreseen limitations are the performance of the internet connection speed and the suppliers' Web Services response time.

- **Software Quality Requirements**

The system should be maintainable and free of defects. The system should achieve a high level of product quality such as reliability and maintainability.

### **4.3 Decisions On The Choices of Development Technologies**

In this section, development technologies such as Web Services platform, web server, database server as well as development language are compared and discussed.

#### **4.3.1 Web Services Development Platform**

Microsoft .NET and Sun Java 2 Enterprise Edition (J2EE) are two of the dominant development platforms that provide the technologies and tools needed to develop and deploy Web Services. From a technical standpoint, each platform has its advantages and disadvantages. Table 4.1 shows a comparison of the .NET framework over the Java Platform (IBM, 2005).

The most important difference between .Net framework and J2EE is concept of independence and interoperability of platform versus language. A .Net component can be written in a combination of languages, e.g. VB.NET and C#. Source code written in these languages is compiled into Microsoft Intermediate Language (MSIL). This MSIL code will be just-in-time compiled to native code at run time by the Common Language Runtime (CLR). However, the only target platform for .NET applications and components to run is Microsoft Windows Operating System.

J2EE, in contrast, is based on Java language. Java source code is compiled to bytecode which is analogous to MSIL code. When the code is ready to run, the Java Virtual Machine interprets this bytecode and executes it at run-time. In theory, any platform with a Java Virtual Machine, including mainframes, Unix systems and Windows, will be able to run the Java applications.

Table 4.1: Comparison of .NET and Java Platforms (IBM, 2005)

Stack Function	.NET	J2EE
Language	C#, VB.NET, C++.NET, other modified languages	Java
Operating System	Microsoft Windows	Multiple Platform
Relational Database Access	ADO.NET	JDBC
Web Client	ASP.NET	Java Server Pages (JSP) and Servlets
Standalone Client	Windows Forms	AWT/Swing
Distributed Components	.NET Remoting	RMI/IDL
XML	System.Xml and .NET in general is built around XML	JAX Pack (JAXM, JAXR, JAXB, JAXP)
Messaging	Microsoft Message Queuing (MSMQ)	Java Messaging Service (JMS)
Web Services Support	Built directly into .NET and Visual Studio.	Java Web Services Developer Pack (JWSDP) as well as vendor specific tools.
Asynchronous Invocation	COM+	Enterprise Java Beans (EJB)
Integration	Host Integration Server, BizTalk Server	J2EE Connector Architecture

.NET framework and Visual Studio.NET have build in support for Web Services which eases the development of building Web Services application. Besides, existing Microsoft .NET skill set is also a key factor that should put into the consideration when choosing the development platform. These factors weigh in the decision to choose the Microsoft .NET as the Web Services development platform.



#### **4.3.2 Web Server**

Two of the most popular web servers are Apache Server and Microsoft Internet Information Service (IIS). However, Apache Web Server currently does not support .NET framework. Thus, Microsoft IIS is the only and natural choice for .Net Web Services to run.

#### **4.3.3 Database Server**

Microsoft SQL Server is chosen as the database server since it is the easier to configure and integrated well with the Microsoft .NET Framework. There are various versions of Microsoft SQL Server, and the version chosen is SQL Server 2000 Enterprise Edition.

#### **4.3.4 Development Language**

One of the key design points for the .NET Common Language Runtime was the support for different programming languages. Microsoft offers a rich palette of four programming languages, namely VB.NET, C#, Visual C++.NET and J#, which together provide developers with the functionality necessary to build robust .NET-connected application. Each language contains unique features and benefits that make it best-suited for certain kinds of applications. VB.NET and ASP.NET has been chosen as the language for the WSEP System due to the simplicity and personal programming experience of Visual Basic.

### **4.4 Hardware Requirements**

The hardware requirements of WSEP system highly depend on the Microsoft .NET Framework that make up the system architecture. Table 4.2 and Table 4.3 below show the minimum system requirements for running the Microsoft .NET Framework 1.1

SDK and the .NET Framework 1.1 Redistributable respectively (Microsoft .NET, 2005).

*Table 4.2: The minimum requirements to install the .NET Framework 1.1 SDK:*

Minimum Requirements	
Processor	Client: 90-megahertz (MHz) Intel Pentium-class processor, or an AMD Opteron, AMD Athlon64 or AMD Athlon XP processor Server: 133-MHz Intel Pentium-class processor, or an AMD Opteron, AMD Athlon64 or AMD Athlon XP processor
Operating System	The .NET Framework 1.1 SDK can be installed on the following platforms: Microsoft Windows® Server 2003 family Windows 2000, with the latest Windows service pack and critical updates. Windows XP (Windows XP Professional is required to run ASP.NET) <b>Note:</b> The .NET Framework SDK 1.1 cannot be installed on 64-bit computers; Windows Millennium Edition and Microsoft Windows NT® 4.0 are not supported
Memory	Client: 32 megabytes (MB) of RAM, 96 MB recommended Server: 128 MB of RAM, 256 MB recommended
Hard Disk	660 MB of hard disk space required, 190 MB additional hard disk space required for installation (850 MB total)
Display	800 x 600 or higher-resolution display with 256 colors
Input Device	Microsoft mouse or compatible pointing device
Other	Prior to installing the .NET Framework SDK version 1.1, you must first install the .NET Framework 1.1 Redistributable Microsoft Internet Explorer 5.01 or later is required Microsoft Data Access Components 2.6 is required for data scenarios. Microsoft Data Access Components 2.8 is recommended on a server.

*Table 4.3 :The minimum requirements to run Web Services on the .NET Framework:*

Minimum Requirements	
Processor	<p>Client (a computer not working in a server capacity): 90-megahertz (MHz) Intel Pentium-class processor, or an AMD Opteron, AMD Athlon64 or AMD Athlon XP processor</p> <p>Server (a computer working in a server capacity): 133-MHz Intel Pentium-class processor, or an AMD Opteron, AMD Athlon64 or AMD Athlon XP processor</p>
Operating System	<p>The .NET Framework 1.1 Redistributable is supported on the following platforms:</p> <p>Microsoft Windows® Server 2003 (.NET Framework 1.1 is installed as part of the operating system)</p> <p>Windows XP Professional</p> <p>Windows XP Home Edition</p> <p>Windows 2000</p> <p>Windows Millennium Edition (Windows Me)</p> <p>Windows 98</p> <p>Microsoft Windows NT® 4.0 Service Pack 6a</p> <p><b>Notes:</b> ASP.NET Web applications and XML Web services can only be hosted on Windows XP Professional, Windows 2000, and Windows Server 2003</p> <p>The .NET Framework 1.1 Redistributable cannot be installed on 64-bit computers; Windows NT 4.0 Terminal Server is not supported</p>
Memory	<p>Client: 32 megabytes (MB) of RAM, 96 MB recommended</p> <p>Server: 128 MB of RAM, 256 MB recommended</p>
Hard Disk	<p>110 MB of hard disk space required, 40 MB additional hard disk space required for installation (150 MB total)</p>

Display	800 x 600 or higher-resolution display with 256 colors
Input Device	Microsoft mouse or compatible pointing device
Other	<p>Microsoft Internet Explorer 5.01 or later is required</p> <p>Microsoft Data Access Components 2.6 is required for data scenarios. Microsoft Data Access Components 2.8 is recommended on a server).</p> <p>Installation of the .NET Framework 1.1 is split into two parts: the core and language packs. The core contains everything you need to run .NET Framework applications; all dialog boxes and error messages will be in English.</p>

## **CHAPTER 5: SYSTEM DESIGN**

System design is a phase that emphasizes on how the system should perform in order to fulfill the requirements identified in the analysis phase. Unified Modeling Language (UML) modeling, such as Sequence Diagrams along with Class Diagrams, are in few of the important design-level models for application development and have been used in this project to specify and document the system.

This chapter details the system architecture and system design for the Web Services E-Procurement System that fully exploits the dynamic characteristic of the Web Services. In general, the system design includes the architecture design, objects and classes design, database design and user interface design.

### **5.1 Web Services E-Procurement Architectural Design**

Software architecture is the high-level structure of a software system. Software architecture is commonly defined in terms of modules, components and connectors. The initial process of architectural design is to identify the main modules that build up the system. Each identified module is then decomposed into a few communicating components with well-defined interfaces. Each component is assigned responsibilities to perform certain functions. A framework has to be established to control these modules and the communication among the modules, and the interactions of components in each module also have to be defined.

The Web Services E-Procurement System (WSEP) consists of 4 modules, which are Authentication Module, Maintenance Module, E-Procurement Transaction Web Services Module and Suppliers' Web Services, as depicts in Figure 5.1. Each module will be described in details in the following sub sections.

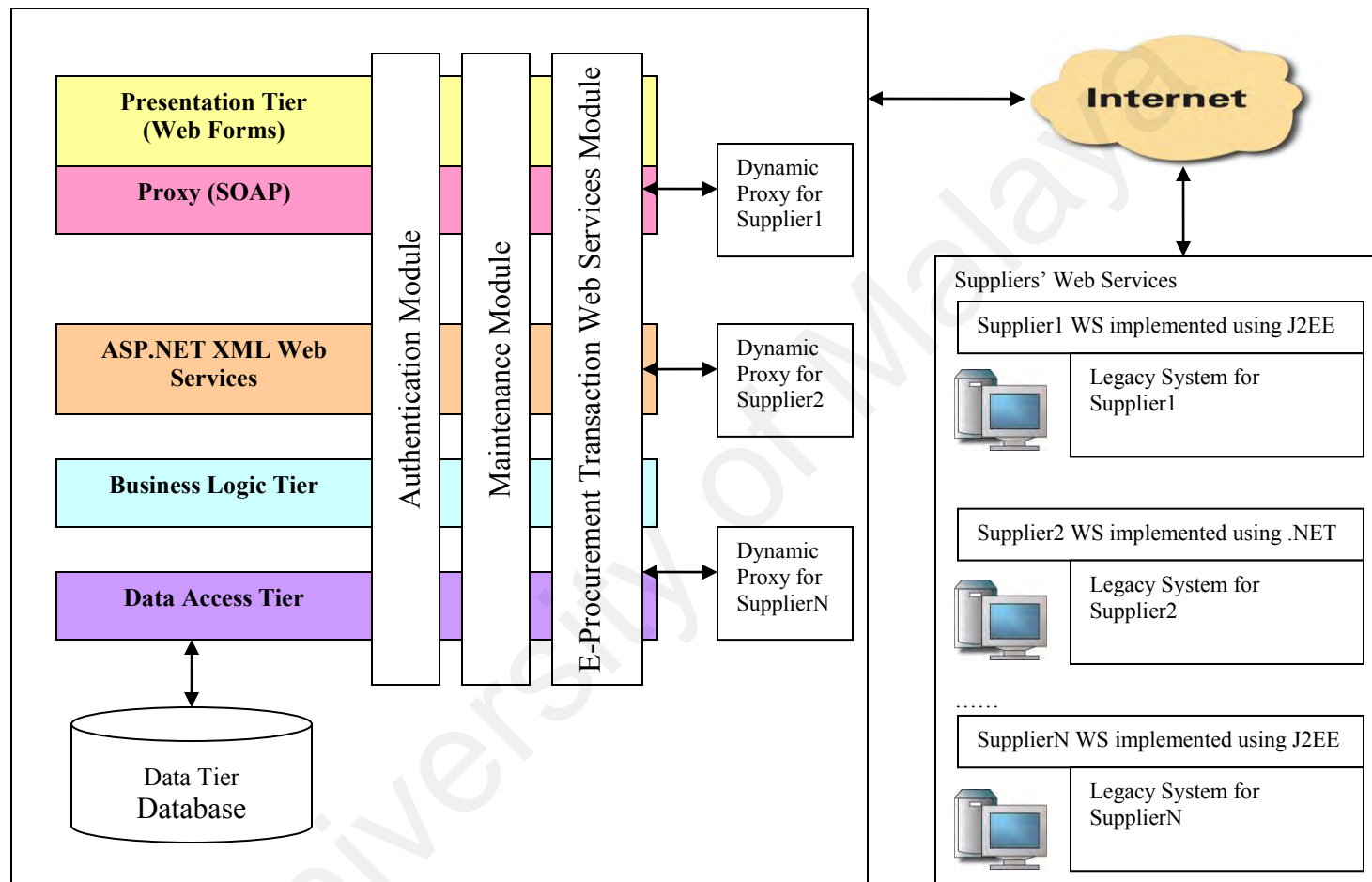


Figure 5.1 Proposed Architecture of Web Services E-Procurement (WSEP) System

### **5.1.1 Authentication Module**

This module is to validate the identity of a user to allow or deny the access to the system. ASP.NET provides built-in support for user authentication through several authentication providers (Kercher, 2001): Forms-based authentication, Microsoft Passport authentication and Windows authentication. This project has used the Form-based authentication. User has to provide the username and password in order for the system to validate against the database.

### **5.1.2 Maintenance Module**

This module is for the Purchase Officer to maintain the entities of the system, which include suppliers, products and suppliers' Web Services. This module performs CRUD (Create, Read, Update and Delete) operations against the entities in the database.

### **5.1.3 E-Procurement Transaction Web Services Module**

The E-Procurement Transaction Web Services is a module implemented using .NET XML Web Services and Reflection. The main function of this module is to bind the supplier Web Services and handle the real-time transactions. It acts as the middle-tier component and provides the necessary business logic rules for procurement processes.

Basically, this module will receive user events from user input. Based on the user selection, this module will query the database to retrieve the suppliers' Web Services URI addresses. It will then dynamically create a proxy to consume the supplier Web Services for information retrieval or transaction.

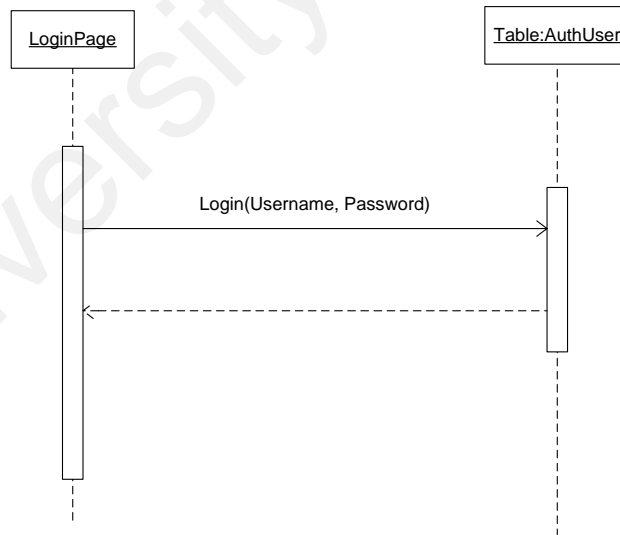
### **5.1.4 Suppliers' Web Services**

Suppliers' Web Services Module is considered as an external module of the WSEP System. Suppliers' Web Services are implemented by the suppliers who want to

participate in the WSEP System. Suppliers have to adhere to the predefined WSDL specifications as shown in Appendix A. The Suppliers' Web Services provide an interface for the WSEP to access the supplier internal information such as product price and quantity on hand. The internal implementations of the Suppliers' Web Services are not important; they can be implemented using .NET, J2EE or other Web Services platforms as long as they conform to the predefined WSDL specifications.

## 5.2 Web Services E-Procurement System Functionality Design

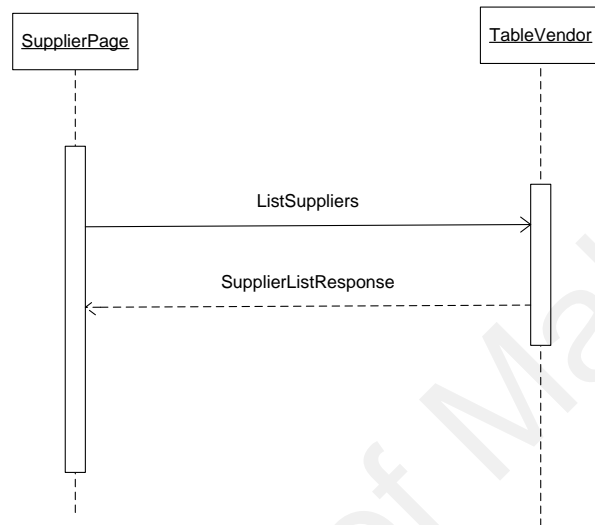
UML Sequence Diagram is used to model the flow of logic within the WSEP system. It helps in identifying the behavior within the system and validating the business logic and interfaces by describing the sequence of actions that need to be performed to complete a task or scenario. Figure 5.2 to Figure 5.16 show all the Sequence Diagrams that correspond to the use cases in section 4.1.1.



*Figure 5.2: Sequence Diagram for User Login*

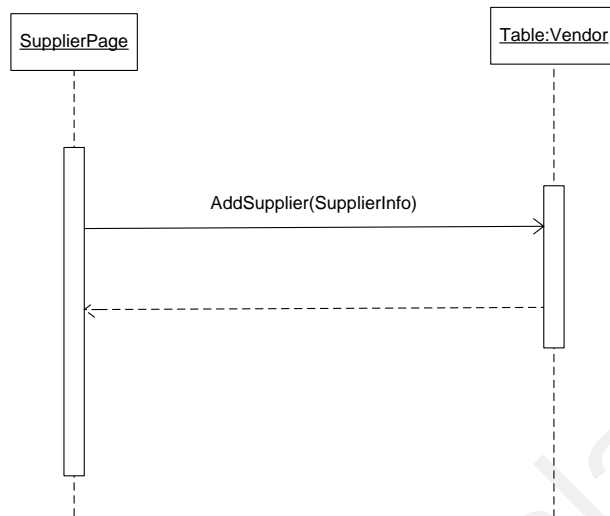


The Purchase Officer provides the username and password through the LoginPage. The LoginPage will query the Table AuthUser in database to validate the information. A successful or failure message will be returned back to the LoginPage to indicate if the Purchase Officer is authorized to access the system.



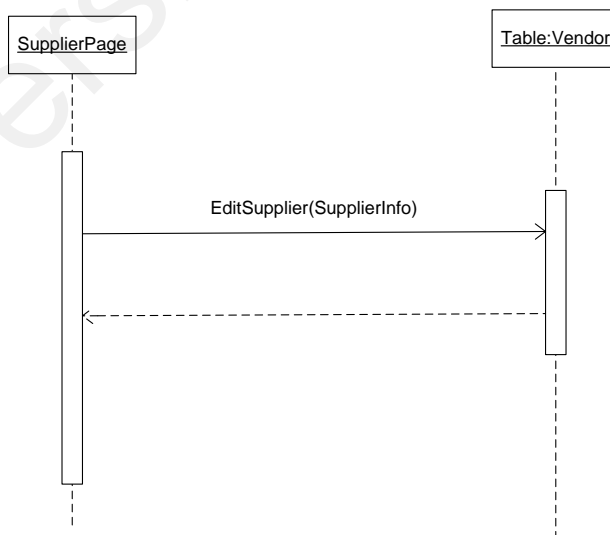
*Figure 5.3: Sequence Diagram for List Suppliers*

The Purchase Officer lists the suppliers through SupplierPage. The SupplierPage will query the Table Vendor in database to get information. The returned results are displayed to the user in grid format.



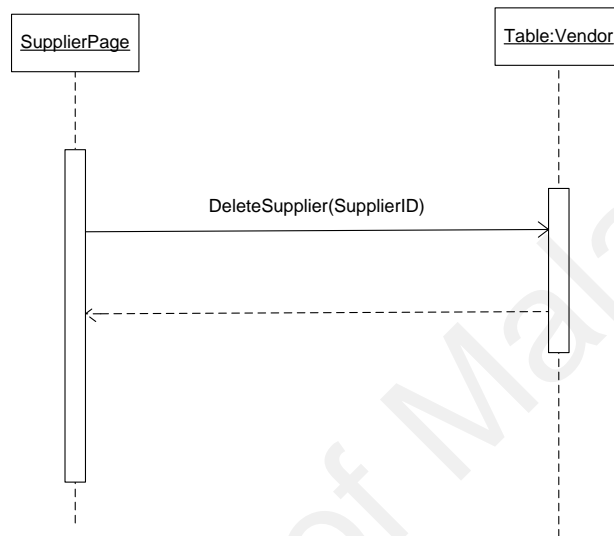
*Figure 5.4: Sequence Diagram for Add Supplier*

The Purchase Officer enters supplier information. The information is submitted to the Table Vendor in database. A successful or failure message will be returned back to notify the Purchase Officer.



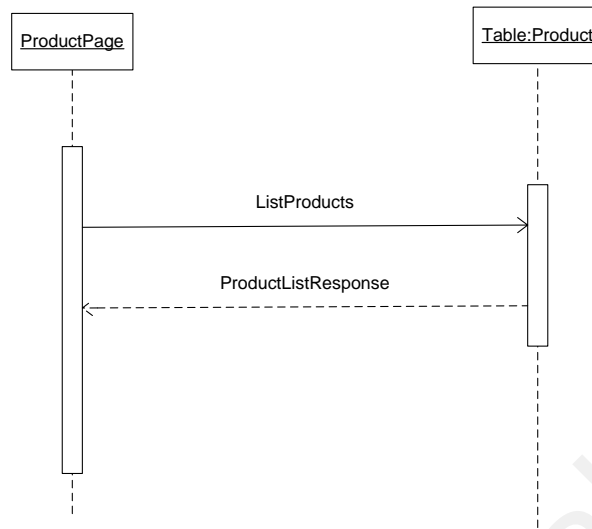
*Figure 5.5: Sequence Diagram for Edit Supplier*

The Purchase Officer edits supplier information. The information is submitted to the Table Vendor in database. A successful or failure message will be returned back to notify the Purchase Officer.



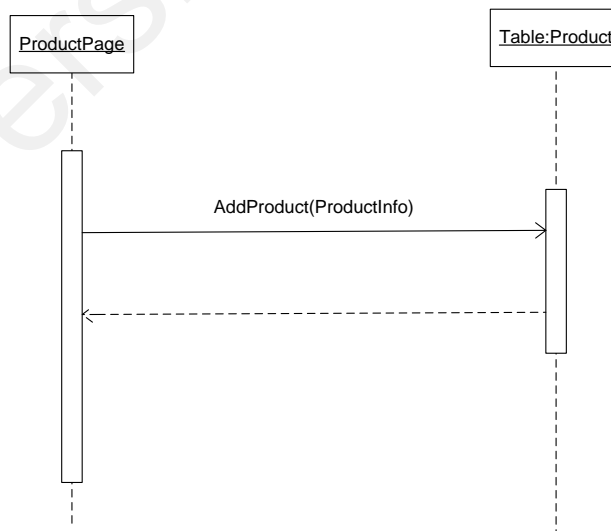
*Figure 5.6: Sequence Diagram for Delete Supplier*

The Purchase Officer deletes the supplier based on SupplierID through SupplierPage. The SupplierPage will delete the record in Table Vendor in database based on the SupplierID. A successful or failure message will be returned back to notify the Purchase Officer.



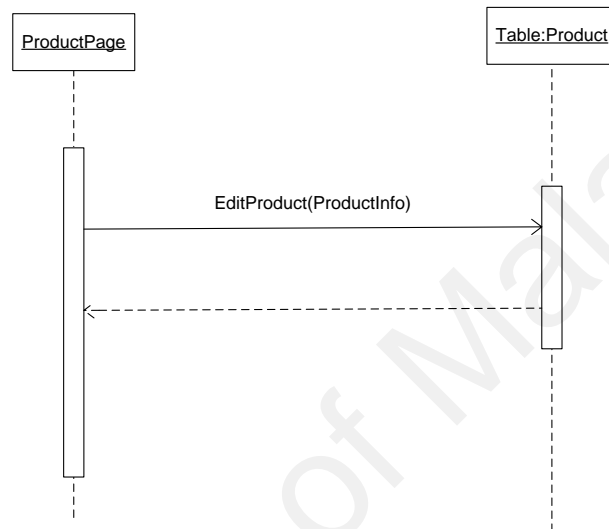
*Figure 5.7: Sequence Diagram for List Products*

The Purchase Officer lists the products through ProductPage. The ProductPage will query the Table Product in database to get information. The returned results are displayed to the user in grid format.



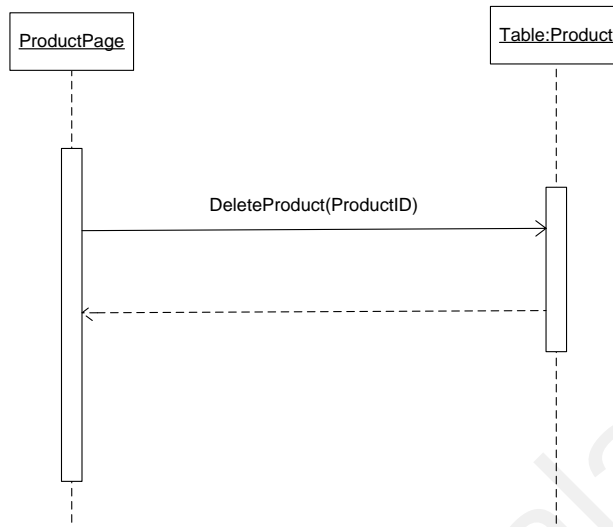
*Figure 5.8: Sequence Diagram for Add Product*

The Purchase Officer enters product information. The information is submitted to the Table Product in database. A successful or failure message will be returned back to notify the Purchase Officer.



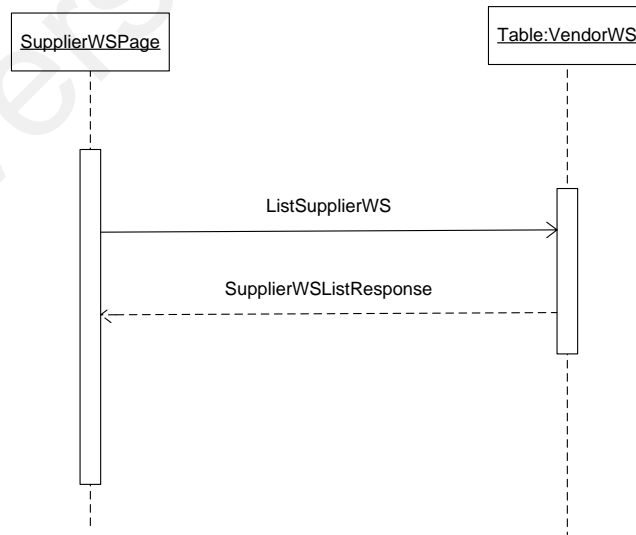
*Figure 5.9: Sequence Diagram for Edit Product*

The Purchase Officer edits product information. The information is submitted to the Table Product in database. A successful or failure message will be returned back to notify the Purchase Officer.



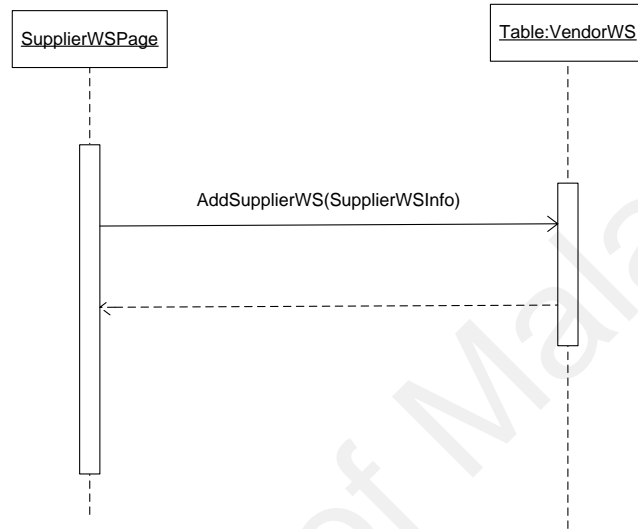
*Figure 5.10: Sequence Diagram for Delete Product*

The Purchase Officer deletes the product based on ProductID through ProductPage. The ProductPage will delete the record in Table Product in database based on the ProductID. A successful or failure message will be returned back to notify the Purchase Officer.



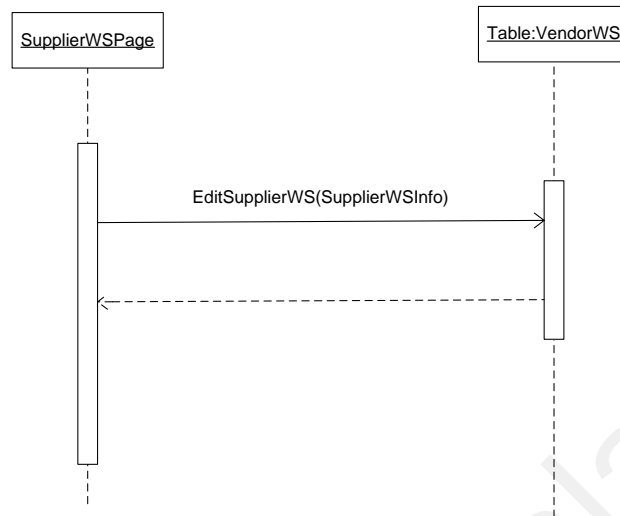
*Figure 5.11: Sequence Diagram for List Supplier Web Services URI*

The Purchase Officer lists the supplier Web Services through SupplierWSPage. The SupplierWSPage will query the Table VendorWS in database to get information. The returned results are displayed to the user in grid format.



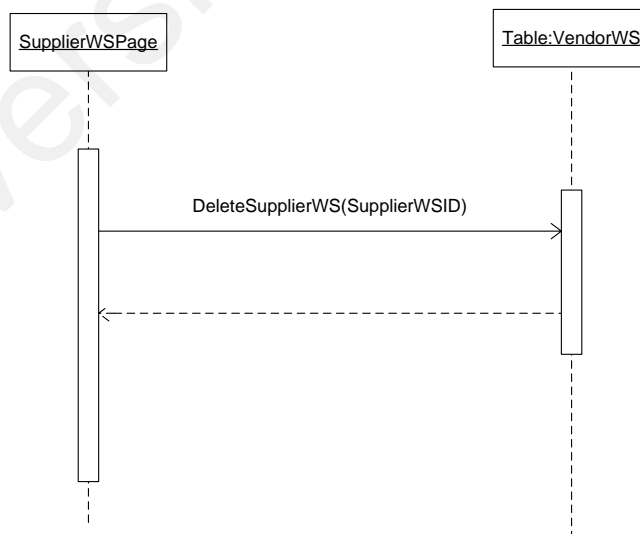
*Figure 5.12: Sequence Diagram for Add Supplier Web Services URI*

The Purchase Officer enters supplier Web Services information. The information is submitted to the Table VendorWS in database. A successful or failure message will be returned back to notify the Purchase Officer.



*Figure 5.13: Sequence Diagram for Edit Supplier Web Services URI*

The Purchase Officer edits supplier Web Services information. The information is submitted to the Table VendorWS in database. A successful or failure message will be returned back to notify the Purchase Officer.



*Figure 5.14: Sequence Diagram for Delete Supplier Web Services URI*



The Purchase Officer deletes the supplier Web Services based on SupplierWSID through SupplierWSPage. The SupplierWSPage will delete the record in Table VendorWS in database based on the SupplierWSID. A successful or failure message will be returned back to notify the Purchase Officer.

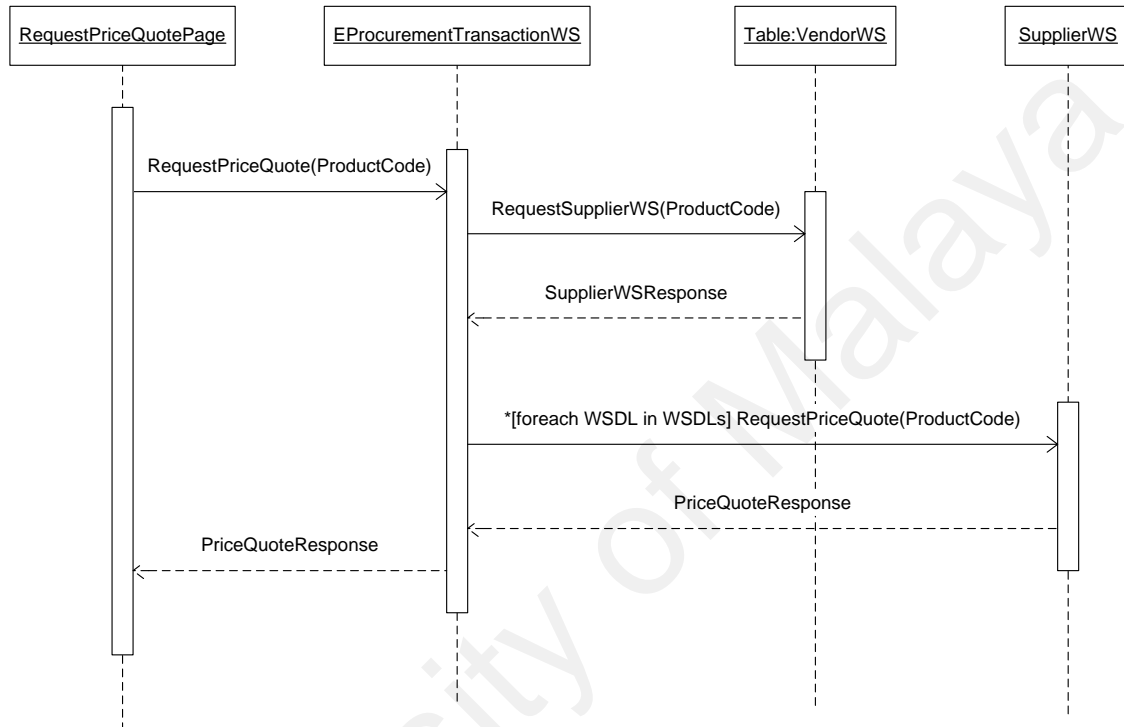


Figure 5.15: Sequence Diagram for Request for Quotation

The process starts with the user initiates a request for a product quotation. The user enters the Product Code in the Quotation Request form and submits to the E-ProcurementTransactionWS module. The E-ProcurementTransactionWS, which is implemented as Web Services, will retrieve the Supplier WSDLs from the Database based on the Product Code. For each WSDL retrieved, the module will send a Quotation Request in XML format to the respective Supplier Web Services. Based on the request, the Supplier Web Services will retrieve the product price along with the quantity on hand from their legacy system and return back the information to the E-ProcurementTransactionWS. The returned results are displayed back to the user in grid format.

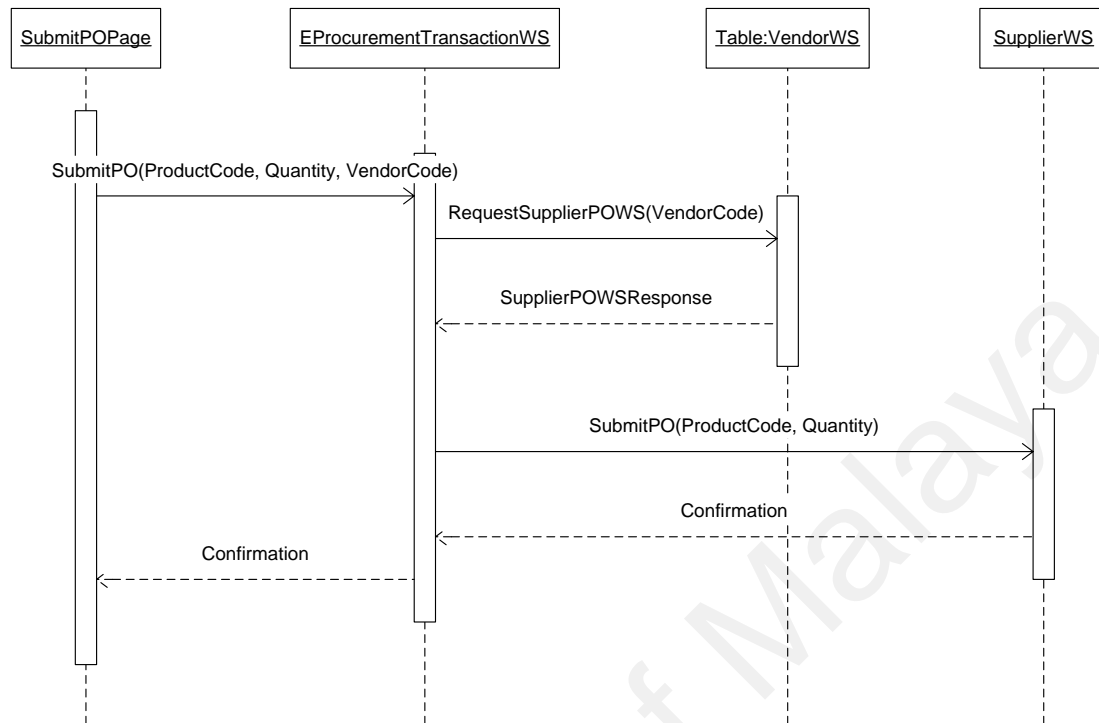


Figure 5.16: Sequence Diagram for Raise Purchase Order

From the results returned, user can select multiple suppliers to place purchase order with different quantity. Again, the E-ProcurementTransactionWS will retrieve the Supplier Web Services information from Database in order to send the Purchase Order to the selected Suppliers. A confirmation of success or failure of the transaction will be sent back to the user. This ends the E-Procurement process.

### 5.3 The Design of Dynamic Aspect of WSEP System

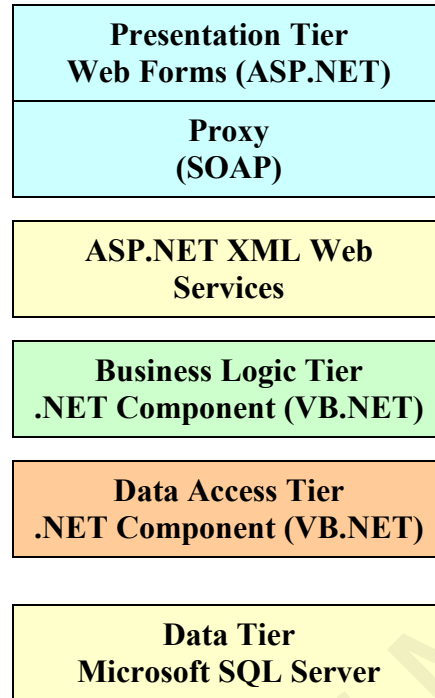
The process of *Request for Quotation* as shown in Figure 5.15 is happened in real time and dynamic manner where the product information, such as pricing and inventory level, is retrieved directly from the supplier inventory systems by interfacing with the Supplier Web Services. Therefore, changes made in the supplier inventory system will be reflected in the quotation during the process.

The result of the process of *Request for Quotation* will be shown to the purchasing officer through the QuotationPage. Once the purchase decision is made, purchase orders will be submitted to the selected suppliers through the E-Procurement Transaction Web Services module. Figure 5.16 shows the dynamic processing of purchase orders in the supply chain. The Web Services URI and WSDL of the supplier are retrieved from the WSEP system database dynamically and the submission of the purchase order is happened in real time.

As shown in Figure 5.1, for each of the selected suppliers, a dynamic proxy is created at run time to communicate with the Supplier Web Services. A proxy is a class that functions as an interface to another thing. It provides a surrogate or placeholder for another object to control access to it (Gamma, 1995). In this context, a proxy is an object that will bind to the Web Services in order to consume the services provided. In order to create and instantiate object dynamically at run time, this project uses Microsoft .NET reflection. Reflection is the mechanism for examining, manipulating and creating objects dynamically at run time (Gilani et al., 2002). DynamicWebServicesLib is a class library created in this project that uses reflection to provide dynamic object or proxy creation facilities to the system. This utility component will handle all the dynamic binding of supplier Web Services at run time. The implementation of this library will be discussed in details in Section 6.3.

#### **5.4 Web Services E-Procurement System Design**

The system design of the WSEP is based on n-tier architecture approach which consists of Presentation Tier, Business Logic Tier, Data Access Tier and Data Tier. Figure 5.17 depicts the n-tier architecture of the WSEP system.



*Figure 5.17: The n-Tier Architecture of the WSEP System (Based on Chartier R., 2005)*

Data Tier is intended to deal with the storage and retrieval of information. This tier can be as complex and comprehensive as high-end products such as SQL Server and Oracle, or all the way down to the simplistic or structured plain text files, e.g. XML, as well as the engine to read and search these files.

Data Tier of the WSEP System is actually the Database Management System (DBMS) that is implemented using Microsoft SQL 2000 Server. According to Microsoft, the data tier encompasses the database server as a whole which comprised of several interrelated layers: the SQL code, the database design, the data storage components on the physical disk, and the server configuration. Figure 5.18 illustrates all the elements of the Data Tier of Microsoft SQL 2000 Server (Microsoft SQL Server 2000 Resource Kit, 2005). WSEP System uses part of the elements of the Microsoft SQL 2000 Server, which are Tables, Indexes and Disk storage.

The Data Tier			
Data Components	Code Components	Storage Components	Server Components
Physical Structure <ul style="list-style-type: none"> <li>• Indexes</li> <li>• Tables</li> <li>• Indexed Views</li> </ul>	Layer of Abstraction: <ul style="list-style-type: none"> <li>• Stored Procedure</li> <li>• Functions</li> <li>• Standard Views</li> </ul>	Filegroups Disk Storage	sp_configure

*Figure 5.18: Elements of the Data Tier of Microsoft SQL 2000 Server (Microsoft SQL Server 2000 Resource Kit, 2005).*

The Data Access Tier is a separate class library that is designed to interface with the Data Tier. It acts as the relay between the application and the backend data store. All the generic data access-specific methods are placed in this layer. Database CRUD (Create, Read, Update and Delete) operations will be supported by this Data Access Tier.

All the business rules are moved to the Business Logic Tier, which sometimes referred to as the Business Services or Middle Tier. The Business Logic Tier is the core component that provides all the functionality services. Normally, this tier is used directly by Presentation Tier and consumes Data Access Tier to retrieve and manipulate data.

The ASP.NET XML Web Services tier acts as the relay between the Presentation Tier and the Business Logic Tier. It provides an interface for the presentation to consumes the functionality services in the Business Logic Tier.

The Presentation Tier is focused on presenting information and receiving input from users. Due to the easy accessible of web application, the Presentation Tier of the system is implemented using ASP.NET Web Form.

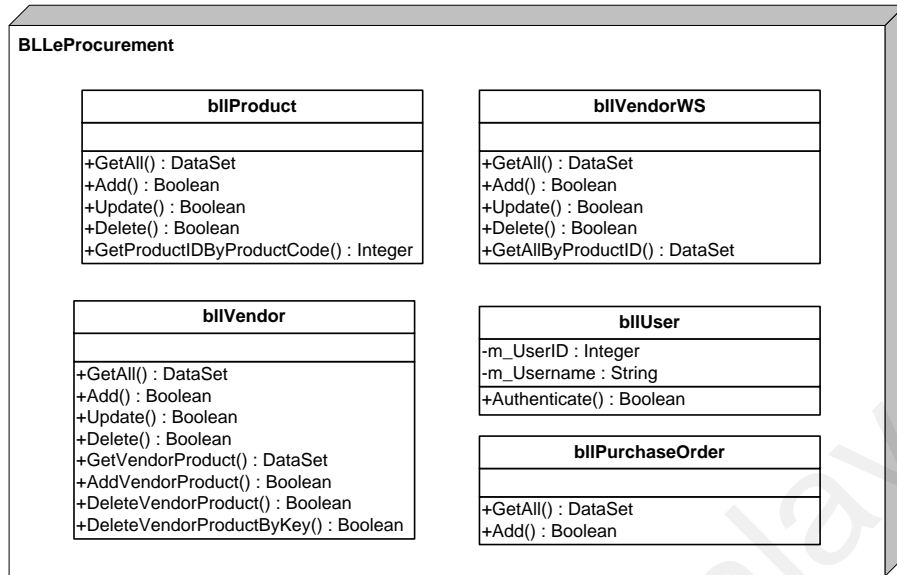
## 5.5 Web Services E-Procurement (WSEP) Components Design

The WSEP system is subdivided into 5 main sub projects:

- DAlLeProcurement, which implements Data Access Tier,
- BLLeProcurement, which implements Business Logic Tier,
- Web eProcurement, which provides the User Interface interacting with the users,
- WSeProcurement, which is the Web Services,
- DynamicWebServicesLib, which is a class library providing the dynamic binding facilities to the system.

Class diagrams give an overview of the system by showing its classes, operations, attributes and the relationships, including inheritance, aggregation and association, among them. UML class diagram is used in this project to model the static design view of the WSEP System.

The classes for the business logic component and data access component as well as the class library component have to be identified during components design. With UML, these classes and components are modeled using class diagram to visualize the static aspects of the building blocks and their relationships and to specify the details for developments, as shown in Figure 5.19, Figure 5.20 and Figure 5.21.



*Figure 5.19: Deployment Diagram for BLLeProcurement*

The BLLeProcurement consists of 5 main classes, i.e. blIProduct, blIVendor, blIVendorWS, blIUser and blIPurchaseOrder. Each of which is responsible for the business rules of the system for their respective module. There is no direct relationship among the classes in the BLLeProcurement.

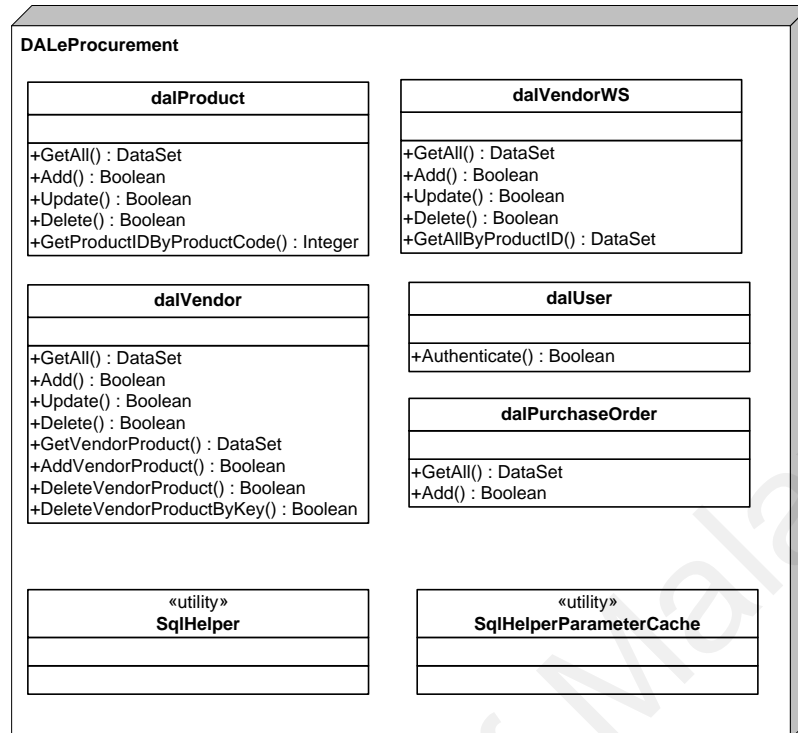


Figure 5.20: Deployment Diagram for DLeProcurement

The DLeProcurement consists of 7 main classes, i.e. dalProduct, dalVendor, dalVendorWS, dalUser and dalPurchaseOrder, SqlHelper and SqlHelperParameterCache. These classes are responsible for accessing the database. SqlHelper and SqlHelperParameterCache are the open source utility classes that used by other DAL classes to facilitate the process of querying database.

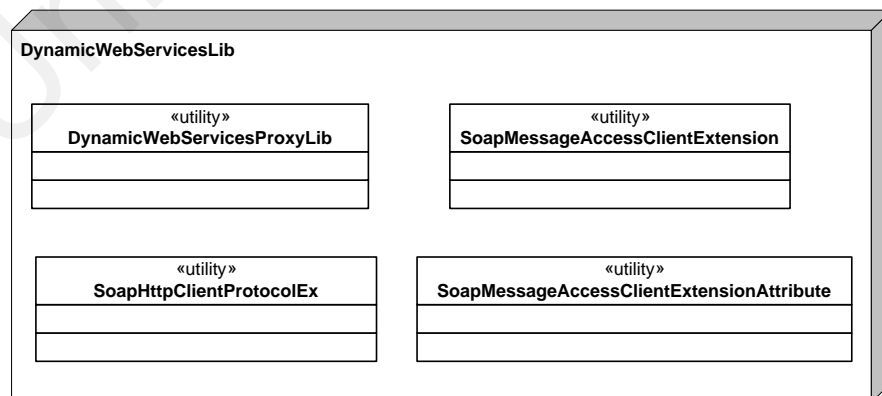


Figure 5.21: Deployment Diagram for DynamicWebServicesLib



The DynamicWebServicesLib is a class library that provides utilities for binding suppliers' Web Services dynamically. This library would be mainly used by WSeProcurement as helper classes to construct a dynamic proxy to invoke Supplier Web Services. The class DynamicWebServicesProxyLib uses .NET reflection to dynamically load assemblies by specifying assembly name.

## **5.6 Web Services E-Procurement Database Design**

Database is used in most of the information systems to store data. A Database Management System (DBMS) is used for creating, updating and modifying and retrieving data. The following sub-sections discuss about the structure and the design of the database of the WSEP system.

### **5.6.1 Database Diagram**

The information of the WSEP system is stored in a relational database. The selected DBMS is Microsoft SQL 2000 Server, which can be managed using the SQL Server Enterprise Manager. The database design of the system is presented in Figure 5.22 below. These tables are in Third Normal Form. Normalization is a formal technique for analyzing relations based on their primary key (or candidate keys) and functional dependencies. This is done to avoid data redundancy and update anomalies. The Third Normal Form (3NF) is a relation that is in first and second normal form, and in which no non-primary key attribute is transitively dependent on the primary key (Thomas and Carolyn, 2002).

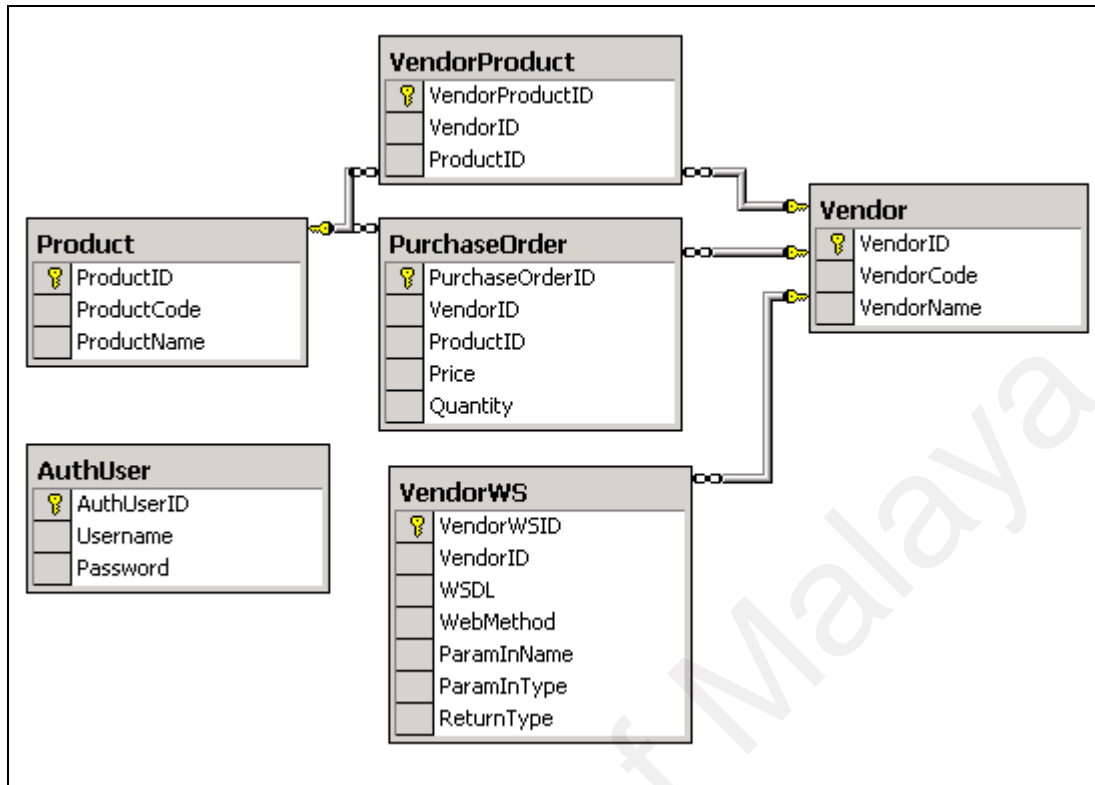


Figure 5.22: Database design of the WSEP System

The AuthUser table is used to store username and password for authentication purpose. The Product table and Vendor table are the entity types that keep product information and supplier information respectively. The VendorProduct table is the entity that provides associative relationship between Vendor and Product. It tells what products are provided by the suppliers. The VendorWS table is used to keep information for the Supplier Web Services. The WSDL for accessing the Supplier Web Services is stored here. For all the purchase orders made, records are kept in the PurchaseOrder table.

### 5.6.2 Data Dictionary

The data dictionary is the repository of all the elements in a system. The following tables are normalized to the Third Normal Form. Table 5.1 shows the table structure of the database. PK represents the Primary Key of the table, whereas FK represents one or more Foreign Keys.

Table 5.1 Table Structure of WSEP System

Table	Attribute	Index	Data Type	Length	Allow Null	Notes
AuthUser	<b>AuthUserID</b>	PK	Int	4	No	Auto-Increment
AuthUser	<b>Username</b>		Varchar	32	No	
AuthUser	<b>Password</b>		Varchar	32	No	
Product	<b>ProductID</b>	PK	Int	4	No	Auto-Increment
Product	<b>ProductCode</b>		Varchar	16	No	
Product	<b>ProductName</b>		Varchar	32	No	
Vendor	<b>VendorID</b>	PK	Int	4	No	Auto-Increment
Vendor	<b>VendorCode</b>		Varchar	16	No	
Vendor	<b>VendorName</b>		Varchar	32	No	
VendorProduct	<b>VendorProductID</b>	PK	Int	4	No	Auto-Increment
VendorProduct	<b>VendorID</b>	FK	Int	4	No	
VendorProduct	<b>ProductID</b>	FK	Int	4	No	
VendorWS	<b>VendorWSID</b>	PK	Int	4	No	Auto-Increment
VendorWS	<b>VendorID</b>	FK	Int	4	No	
VendorWS	<b>WSDL</b>		Varchar	128	No	
VendorWS	<b>WebMethod</b>		Varchar	64	No	
VendorWS	<b>ParamInName</b>		Varchar	64	No	
VendorWS	<b>ParamInType</b>		Varchar	32	No	
VendorWS	<b>ReturnType</b>		Varchar	32	No	
PurchaseOrder	<b>VendorProductID</b>	PK	Int	4	No	Auto-Increment
PurchaseOrder	<b>VendorID</b>	FK	Int	4	No	
PurchaseOrder	<b>ProductID</b>	FK	Int	4	No	
PurchaseOrder	<b>Price</b>		Decimal	9	No	
PurchaseOrder	<b>Quantity</b>		Int	4	No	

## 5.7 User Interface Design

User interface design must take into account the needs, experience and capabilities of the system users. Table 5.2 lists the design principles (Ian, 2001) that are incorporated into the user interface design of the WSEP system.

*Table 5.2: User interface design principles (Ian, 2001)*

Principle	Description
User familiarity	The interface should use terms and concepts which are drawn from the experience of the anticipated class of user.
Consistency	The interface should be consistent in that comparable operations should be activated in the same way.
Least Surprise	Users should never be surprised by the behavior of the system.
Recoverability	The interface should include mechanisms to allow users to recover from their errors.

- **User familiarity**

Novelty is a barrier to entry. It puts a learning burden on the user. Thus, the interface should use terms and concepts which are drawn from the experience of the anticipated class of user. The user interface for the WSEP System follows the general navigation and layout conventions of major web sites that most users may already be used to those conventions. Figure 5.23 shows one of the screen that uses terms and objects, e.g. ‘Vendors’, ‘Products’, ‘Request For Quote’, ‘Edit’, ‘Delete’ and so on, that have direct analogues in the user’s environment and familiar to the user.



*Figure 5.23: Screen uses terms and objects that have direct analogues in the user's environment and familiar to the user*

- **Consistency**

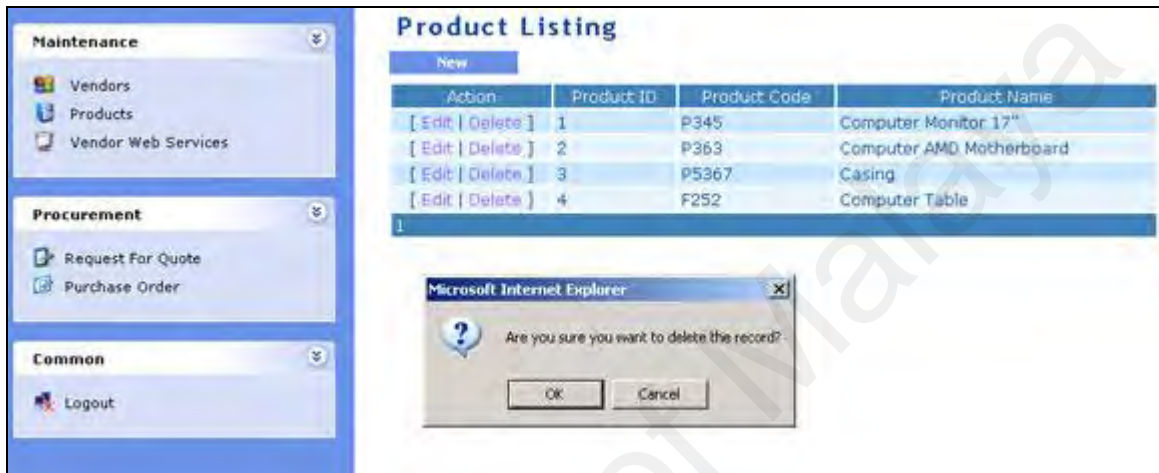
The user interface is built on a consistent pattern across all pages that all share the same navigation menus, basic data grids, graphic themes and layouts. The goal is to be consistent and predictable so that the users would feel comfortable and confident while using the system. The menu consists of 3 parts namely Maintenance, Procurement and Common which presents on every page at the left side panel. The design of the system supports the same operations such as New, Edit and Delete on all types of the system entities.

- **Least Surprise**

As a system is used, users will have a basic concept of the system works. Users will become frustrated and irritated if a system behaves in an unexpected way. The user interface design of the WSEP system is striving towards the principle of "least surprise". If one attribute of an object is changed in a particular way, then it is to be expected that other attributes of the object would also be changed in a similar fashion.

- **Recoverability**

Users may make mistakes when using the system. The WSEP system provides some resilience to user errors and allows the user to recover from errors. This includes undo facility and confirmation of destructive actions as shown in Figure 5.24.



*Figure 5.24: Confirmation of destructive actions*

## 5.8 Conclusion

The system architecture of WSEP is proposed. UML Sequence diagrams are used to model the flow of logic within the WSEP system. In addition, the system classes and components are identified and modeled using UML class diagrams. The design of dynamic and real time aspects of the WSEP system is discussed. The database structure is designed and normalized to Third Normal Form. Some of the design interface principles are incorporated into the user interface design of the WSEP system.

## **CHAPTER 6: SYSTEM IMPLEMENTATION**

In the implementation phase, the system requirements are converted to system source codes. With the requirement documentation and the system design documentation from the previous phase, the system should be built according to what has been documented. The goal of the implementation phase is to implement the system correctly on the particular development environment in accordance with the project plan. The end deliverable of this phase is the system itself.

### **6.1 Development Environment**

#### **6.1.1 Software Development Environment**

Most of the programming languages with which we work require a development environment to code, test and run the programs. For this project, ASP.NET is used as the development technology for Web Services that is built into the .NET Framework. ASP.NET Web Services applications can be created with a simple editor like a notepad or Microsoft integrated development environment (IDE) Visual Studio .NET 2003. Microsoft Visual Studio .NET 2003 is an advanced IDE tool for developing of .NET application. It is used in this project for all the development of Web Application, Web Services and components.

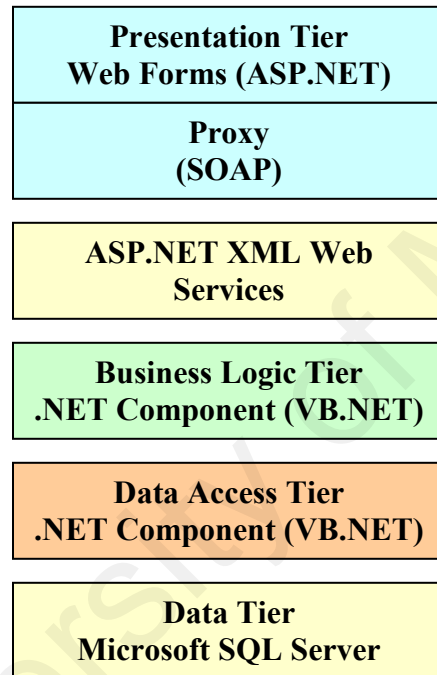
#### **6.1.2 Hardware Development Environment**

The following hardware specifications have been used to develop the WSEP system and also act as the Web Server:

- a. AMD Athlon 2500 ++
- b. 512 MB RAM
- c. 80 GB Hard Disk
- d. Standard hardware for IBM PC compatible machine

## 6.2 System Implementation

The system design of the WSEP is based on n-tier architecture approach. Figure 6.1 shows the architecture consists of Presentation Tier, ASP.NET XML Web Services, Business Logic Tier, Data Access Tier and Data Tier. This section explains the implementation of each tier in details.



*Figure 6.1: The n-Tier Architecture of the WSEP System*

### 6.2.1 Implementation of Data Access Tier

Data Access Tier is a component that interacts with the backend data store. The generic data access-specific methods, i.e. CRUD (Create, Read, Update and Delete) operations will be supported by this component. In this project, an open-source helper class library, Data Access Application Block (DAAB), provided by Microsoft has been



used to streamline the database operations by simplifying the creation of a connection, specifying a command, executing the query and etc.

Figure 6.2 shows a function, GetAll(), that returns a Dataset of the Vendor in one of the class of the Data Access Tier component. First, an SQL string is constructed. The SQL string may consists of WHERE criteria if the pass-in parameter is not null. Then the SQL string is executed against the Database by using ExecuteDataset() static method of the SqlHelper class, provided by DAAB, to fetch the required records from the table. The Globals.ConnectionString is a variable that keeps the credentials for connecting to SQL database.

```
Return SqlHelper.ExecuteDataset(Globals.ConnectionString,  
    CommandType.Text, strSQL)
```

Besides ExecuteDataset() which return a set of data, ExecuteScalar() and ExecuteNonQuery() methods of the SqlHelper class are also used for Updating, Inserting and Deleting records in the Database.

```

Public Function GetAll(ByVal VendorID As Object, ByVal VendorCode As Object,
ByVal VendorName As Object) As DataSet

    Dim strSQL As String = "" ' Used for the sql SELECT clause
    Dim strWHERE As String = "" ' Used for the sql WHERE clause

    ' Build SELECT clause
    strSQL = strSQL & "SELECT [Vendor].[VendorID] "
    strSQL = strSQL & "      , [Vendor].[VendorCode] "
    strSQL = strSQL & "      , [Vendor].[VendorName] "
    strSQL = strSQL & " FROM Vendor "

    ' Build WHERE clause. Only include criteria if parameter is not null.
    If Not VendorID Is DBNull.Value Then strWHERE = strWHERE & " AND
[Vendor].[VendorID] = " & VendorID.ToString()
    If Not VendorCode Is DBNull.Value Then strWHERE = strWHERE & " AND
[Vendor].[VendorCode] = " & SqlHelper.SQLString(VendorCode)
    If Not VendorName Is DBNull.Value Then strWHERE = strWHERE & " AND
[Vendor].[VendorName] = " & SqlHelper.SQLString(VendorName)

    ' Concatenate SELECT and WHERE clauses, removing first AND from WHERE
    clause.
    If strWHERE <> "" Then strSQL = strSQL & " WHERE " &
strWHERE.Substring(4)

    ' Execute the SQL and return the data
    Try
        Return SqlHelper.ExecuteDataset(Globals.ConnectionString,
CommandType.Text, strSQL)

    Catch ex As Exception
        Throw New Exception(ex.Message, ex)
    Finally

    End Try

```

*Figure 6.2: Function GetAll() that returns a Dataset in the Data Access Tier component*

## 6.2.2 Implementation of Business Logic Tier

All the business rules are in Business Logic Tier. Normally, this tier is consumed directly by Presentation Tier and will call Data Access Tier component to retrieve data required. Figure 6.3 shows one of the functions in the Vendor class that calls GetAll() method of dalVendor class in Data Access Tier. This function simply returns the Dataset that is retrieved from the dalVendor class to the Presentation Tier.

Typically, a Business Logic method in this tier can be described as:

- Instantiate an Data Access object
- Retrieve the crude data.
- Calculate business values from the crude data
- Return the data to the Presentation Tier

```
Public Function GetAll(ByVal VendorID As Object, ByVal VendorCode As Object,
ByVal VendorName As Object) As DataSet

    Dim objVendor As New dalVendor

    Try
        Return objVendor.GetAll(VendorID, VendorCode, VendorName)

    Catch ex As Exception

        Throw New Exception(ex.Message, ex)

    Finally

        If Not objVendor Is Nothing Then objVendor = Nothing

    End Try

End Function
```

*Figure 6.3: Function GetAll()in the Business Logic Tier component.*

### **6.2.3 Implementation of the Web Services**

The Web Services are implemented using ASP.NET. Figure 6.4 lists all the web methods or operations supported.



*Figure 6.4: List of all web methods in Web Services*

Basically, the Web Services act as an interface for the Business Logic that will be called by the presentation. Most of Web Services methods will call the Business Logic Layer components which in turn will call the Data Access Layer components to perform data functions. Figure 6.5 shows the GetVendor() web method that calls the GetAll() function in the bllVendor class of the Business Logic Layer component. This method will return a DataSet of vendors that meet the search criteria to the Presentation Layer.

```

<WebMethod()> _
Public Function GetVendor(ByVal VendorID As Integer, _
    ByVal VendorCode As String, _
    ByVal VendorName As String) As DataSet

    Dim objVendor As New bllVendor
    Dim mVendorID As Object
    Dim mVendorCode As Object
    Dim mVendorName As Object

    Try
        If VendorID = 0 Then
            mVendorID = DBNull.Value
        Else
            mVendorID = VendorID
        End If

        If VendorCode = "" Then
            mVendorCode = DBNull.Value
        Else
            mVendorCode = VendorCode
        End If

        If VendorName = "" Then
            mVendorName = DBNull.Value
        Else
            mVendorName = VendorName
        End If

        Return objVendor.GetAll(mVendorID, mVendorCode, mVendorName)

    Catch ex As Exception
        Throw New Exception(ex.Message, ex)
    Finally
        If Not objVendor Is Nothing Then objVendor = Nothing
    End Try
End Function

```

*Figure 6.5: Web Method GetVendor() in the Web Services component*

In order to provide dynamic binding of qualified Suppliers' Web Service, DynamicWebServicesProxyLib library is used to create the proxy at run time that binds the Suppliers' Web Services. The code for the web method RequestForQuotation is listed in Figure 6.4.

```

<WebMethod(>> _
Public Function RequestForQuotation(ByVal ProductCode As String, ByRef
ErrorMessage As String) As DataSet

    Dim dsWS As DataSet
    Dim dsTempQuotation As New DataSet
    Dim dsQuotation As New DataSet
    Dim WSProxy As New DynamicWebServicesProxyLib
    Dim i As Integer
    Dim m As Integer
    Dim ProductID As Integer
    Dim VendorID As Integer
    Dim VendorCode As String
    Dim VendorName As String

    Try

        ProductID = GetProductKey(ProductCode)
        If ProductID > 0 Then

            dsWS = GetVendorWS(ProductID)
            Dim dcVendorID As DataColumn
            Dim dcVendorCode As DataColumn
            Dim dcVendorName As DataColumn

            Dim dt As DataTable = dsWS.Tables(0)

            For i = 0 To dt.Rows.Count - 1
                VendorID = CType(dt.Rows(i).Item("VendorID"), Integer)
                dsTempQuotation = New DataSet

                Try

                    WSProxy.TypeName = "PriceQuote"
                    WSProxy.WSDL = dt.Rows(i).Item("WSDL").ToString
                    WSProxy.MethodName =
dt.Rows(i).Item("WebMethod").ToString
                    WSProxy.AddParameter(ProductCode)

                    dsTempQuotation = CType(WSProxy.InvokeCall, DataSet)

                    If Not (dsTempQuotation Is Nothing) Then

                        dcVendorID = New DataColumn("VendorID",
GetType(Integer))
                        dcVendorCode = New DataColumn("VendorCode" ,
GetType(String))
                        dcVendorName = New DataColumn("VendorName" ,
GetType(String))

```

*Continue ...*

```

        dsTempQuotation.Tables(0).Columns.Add(dcVendorID)
        dsTempQuotation.Tables(0).Columns.Add(dcVendorCode)
        dsTempQuotation.Tables(0).Columns.Add(dcVendorName)

        If GetVendorInfo(VendorID, VendorCode, VendorName)
Then
            For m = 0 To
                dsTempQuotation.Tables(0).Rows.Count - 1

                dsTempQuotation.Tables(0).Rows(m).Item("VendorID") = VendorID

                dsTempQuotation.Tables(0).Rows(m).Item("VendorCode") = VendorCode

                dsTempQuotation.Tables(0).Rows(m).Item("VendorName") = VendorName
            Next
        Else
            For m = 0 To
dsTempQuotation.Tables(0).Rows.Count - 1

                dsTempQuotation.Tables(0).Rows(m).Item("VendorID") = VendorID

                dsTempQuotation.Tables(0).Rows(m).Item("VendorCode") = "Unknown"

                dsTempQuotation.Tables(0).Rows(m).Item("VendorName") = "Unknown"
            Next
        End If
        dsQuotation.Merge(dsTempQuotation)

        'Reset the object
        dsTempQuotation = Nothing

        dcVendorID = Nothing
        dcVendorCode = Nothing
        dcVendorName = Nothing
    End If

```

*Continue ...*

```

        Catch ex As Exception
            If GetVendorInfo(VendorID, VendorCode, VendorName)
Then
                ErrorMessage = ErrorMessage & "There are errors
while processing Web Services for Vendor " & VendorCode & " ( " & VendorName &
" ).<br>"
                ErrorMessage = ErrorMessage & "[Error Message : "
& ex.Message & "]<br><br>"
            Else
                ErrorMessage = ErrorMessage & "There are errors
while processing Web Services for VendorID " & VendorID & ".<br>"
                ErrorMessage = ErrorMessage & "[Error Message : "
& ex.Message & "]<br><br>"
            End If
        End Try
    Next

    RequestForQuotation = dsQuotation

Else
    Return Nothing
End If

Catch ex As Exception
    ErrorMessage = ErrorMessage & ex.Message & " <br><br> "
    Throw New Exception(ErrorMessage, ex)
Finally
    If Not WSPProxy Is Nothing Then WSPProxy = Nothing
End Try
End Function

```

*Figure 6.6: Web Method RequestForQuotation () in the Web Services component*



## 6.2.4 Implementation of WSEP Database

The implementation of WSEP database is straight forward. Based on the Data Dictionary defined in Section 5.5.2, creating tables and relationships among tables can be done easily by using SQL Server Enterprise Manager. Figure 6.7 shows the easy steps to create table Vendor and define Relationship between Table Vendor and VendorWS based on the Primary Key and Foreign Key.

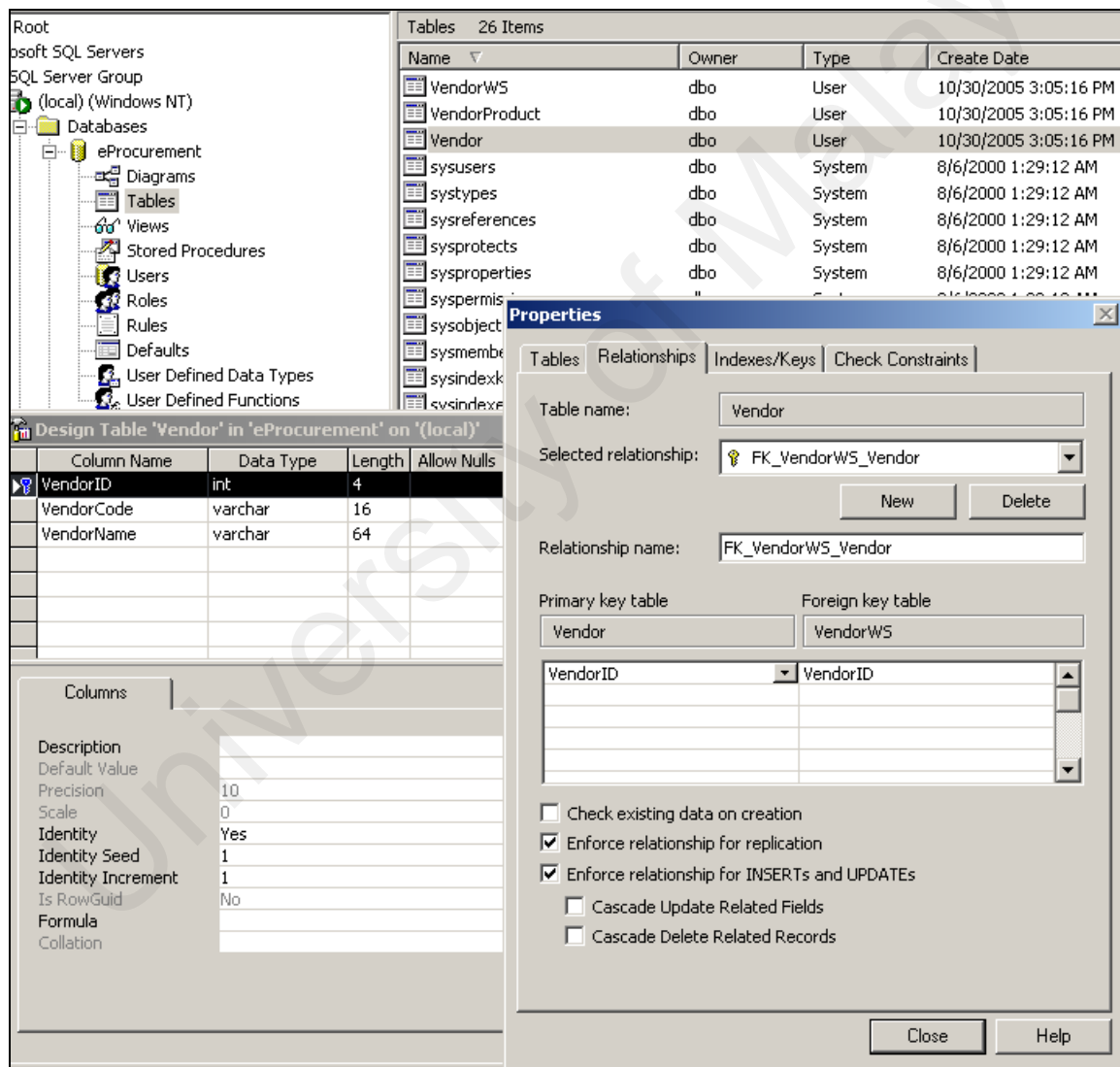


Figure 6.7: Create Table and Relationship using SQL Server Enterprise Manager

By creating a Database Diagram using SQL Server Enterprise Manager, the database schema can be viewed in one glance as depicted in Figure 6.8. This simplifies the changes made to the schema of the database.

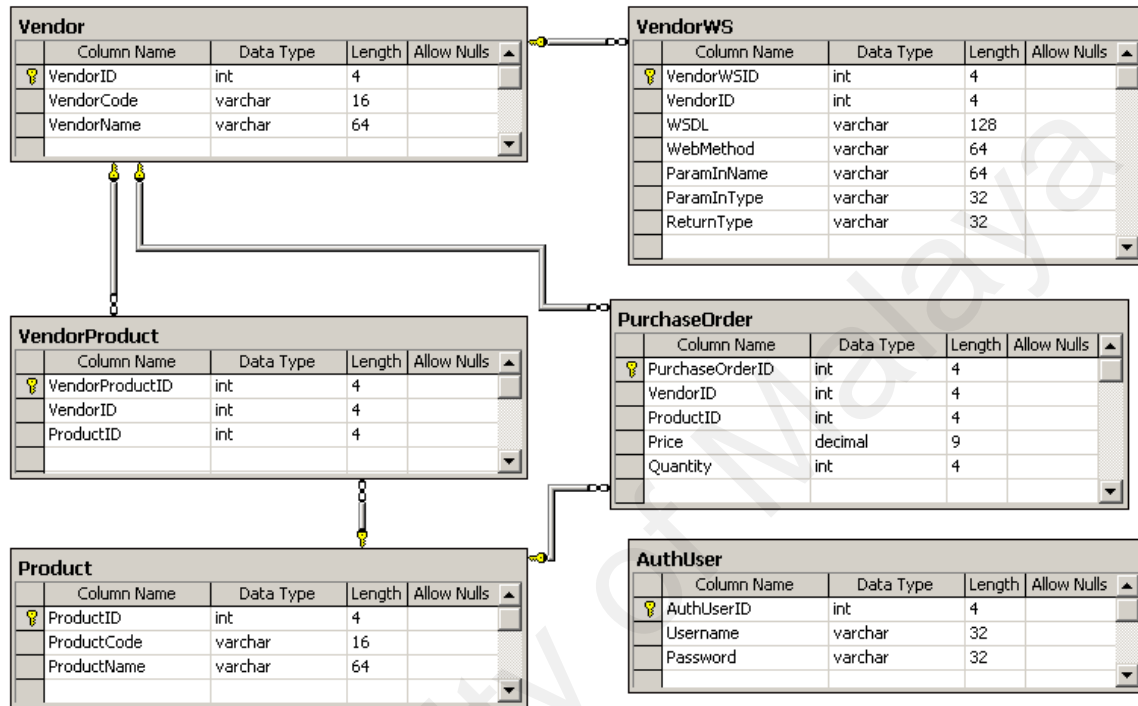


Figure 6.8: Detailed View Database Diagram

### 6.2.5 Implementation of User Interface using ASP.NET

Web Services are accessed by the Web Application by using a proxy class. For static binding, the proxy class can be created automatically by using Visual Studio. This is done by setting up a Web Reference by right-clicking the Web Reference for the project as shown in Figure 6.9. The screen in Figure 6.10 will pop up, giving a choice of sources for the Web services.

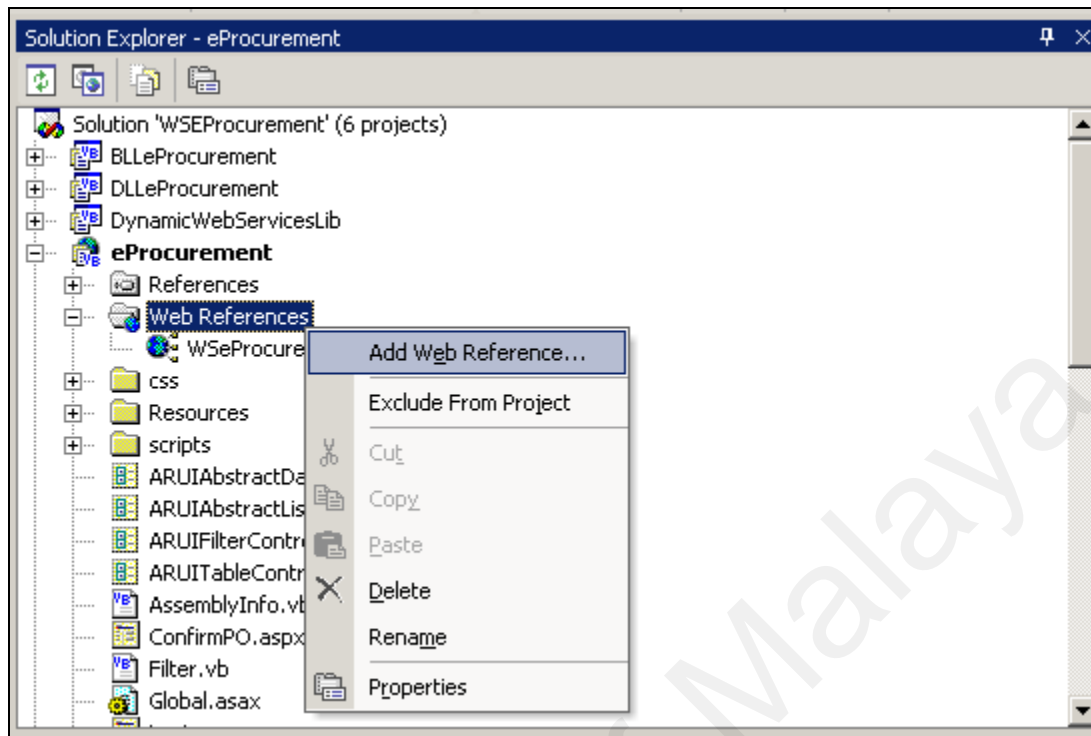


Figure 6.9: Adding Web Reference through Visual Studio 2003

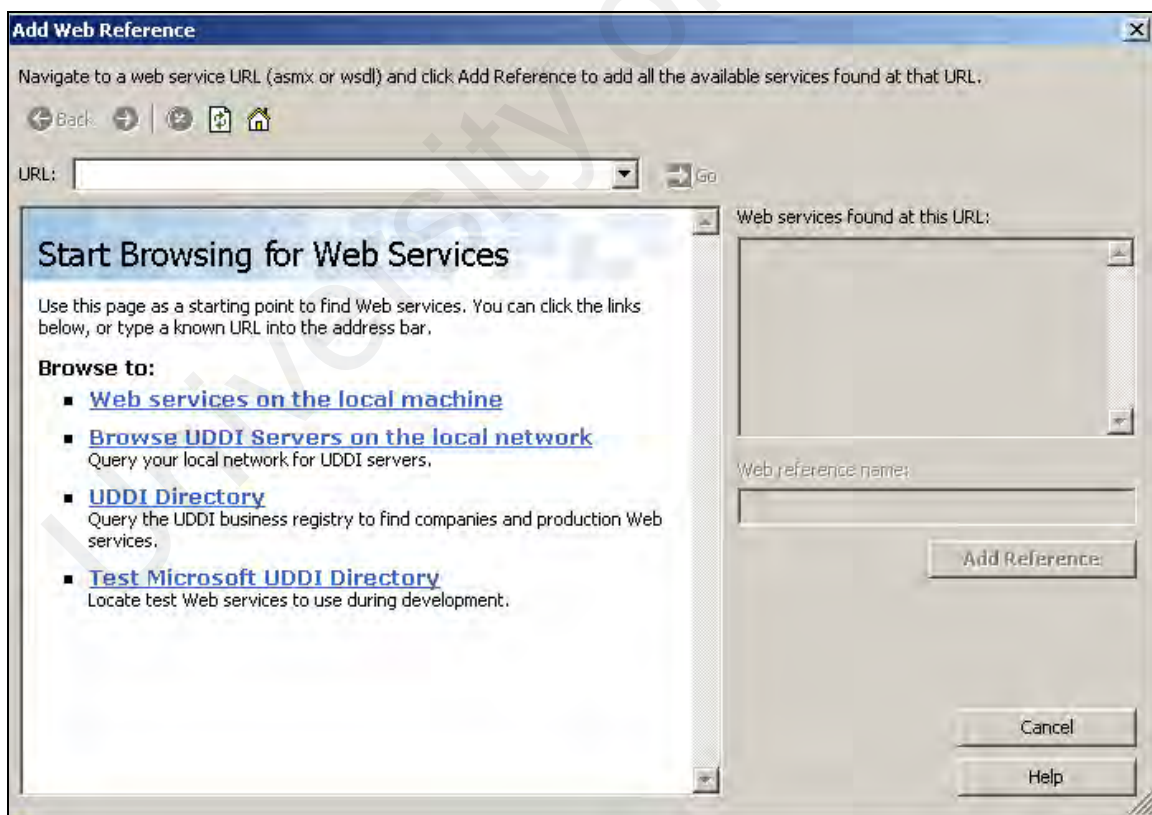


Figure 6.10: Web Services on the Local Machine

Clicking on Web Services on the local machine will bring up a list of Web Services available on the local machine. By selecting the intended Web Services, it will bring up Figure 6.11 that enables final selection of the Web Service. Clicking on the Add Reference will add a Web Reference into the Solution Explorer.

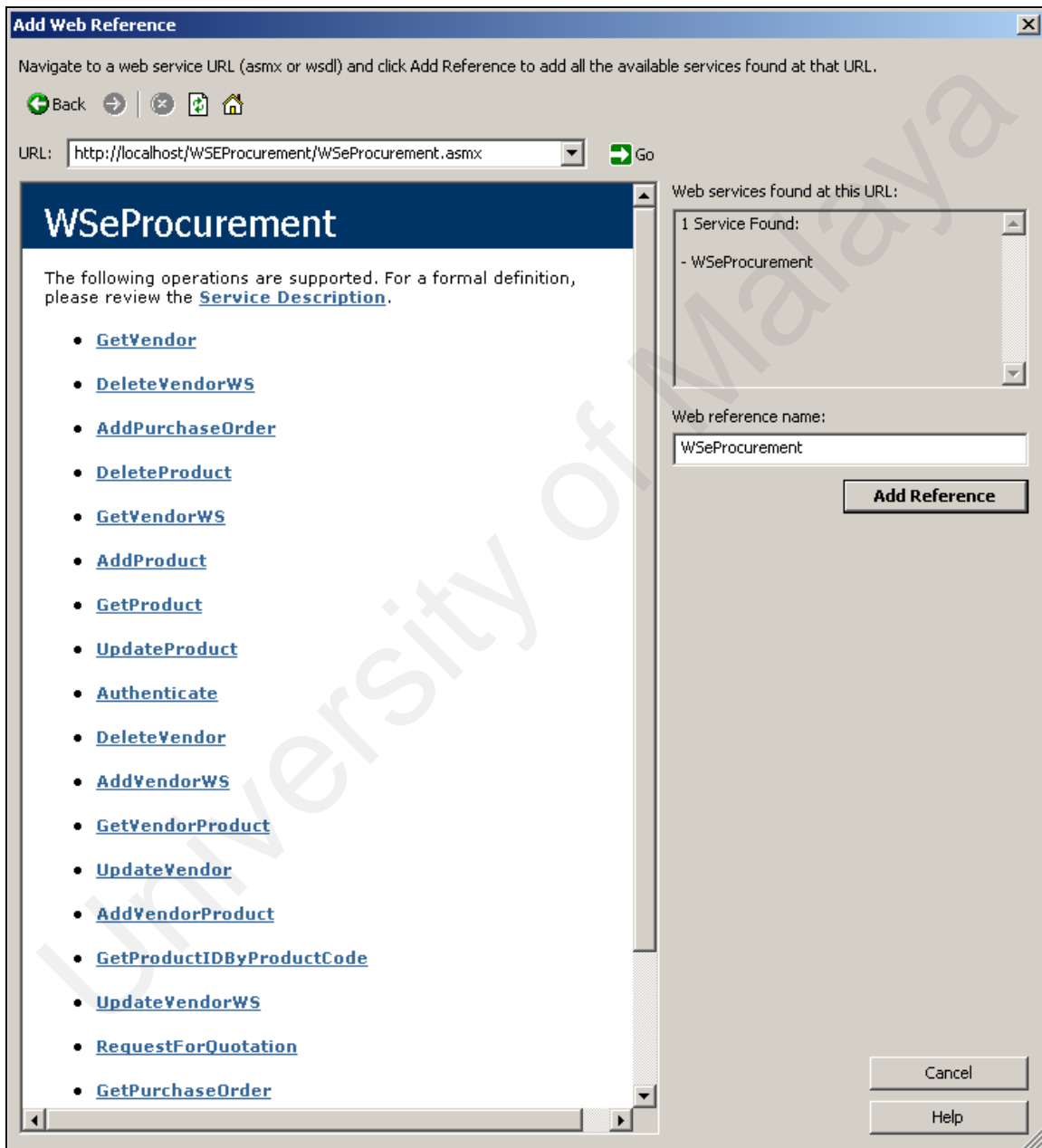


Figure 6.11: Dialog box to Add Reference

### 6.2.6 Implementation of Supplier Web Services

Suppliers' Web Services Module is considered as an external module of the system. The current design of system requires all the participating suppliers to implement Web Services that conform to the standard as defined by WSDL. The Web Services provide an interface for the WSEP to access the supplier internal information such as product price and stock level. By providing the ProductCode, the Web Services method should return a DataSet that contains all the required information for the product. As mentioned earlier, the Suppliers' Web Services can be implemented using .NET, J2EE or other Web Services platforms. Figure 6.12 illustrates a sample of implementation for getting the internal inventory information of the supplier using .NET framework that conformed to the published WSDL.

```

<WebMethod()> _
    Public Function GetInventoryInfo(ByVal ProductCode As String) As DataSet

        Dim conn As New
SqlConnection(ConfigurationSettings.AppSettings("sqlConn"))
        Dim cmd As SqlCommand

        Dim strSQL As String = ""
        strSQL = " SELECT  Price, "
        strSQL = strSQL & " QOH AS QuantityOnHand, "
        strSQL = strSQL & " UOM AS UnitMeasurement "
        strSQL = strSQL & " FROM Inventory "
        strSQL = strSQL & " WHERE WSProductCode =" & ProductCode & " "
        strSQL = strSQL & ProductCode & " "

        Try
            conn.Open()
            cmd = New SqlCommand(strSQL, conn)
            Dim ds As New DataSet
            Dim da As New SqlDataAdapter(cmd)
            da.Fill(ds)

            GetInventoryInfo = ds

        Catch ex As Exception

        Finally
            conn.Close()
            cmd.Dispose()
        End Try

    End Function

```

*Figure 6.12: Sample of Implementation of Supplier Web Services*

### 6.3 Implementation of Dynamic Aspect of WSEP System

Web method RequestForQuotation from the list of Figure 6.4 is not calling the BLL components. Instead, it calls the DynamicWebServicesProxyLib class library to generate proxy in order to connect to the Supplier Web Services. By using .NET Reflection, the proxy or assembly can be generated at runtime.

Figure 6.13 shows part of the codes in web method RequestForQuotation. Based on the Product Code, supplier WSDLs are retrieved from the database. An instance, i.e.

WSProxy, of class DynamicWebServicesProxyLib is declared and instantiated. dsWS is the dataset that contains records retrieved from the table VendorWS in the database. For each of the WSDL retrieved from the table VendorWS, the instance invokes the public function, InvokeCall(), that returns an object which is then converted into a dataset. This dataset is the return result of quotation from the supplier Web Services.

```
Dim WSProxy As New DynamicWebServicesProxyLib
Dim dt As DataTable = dsWS.Tables(0)

For i = 0 To dt.Rows.Count - 1
    VendorID = CType(dt.Rows(i).Item("VendorID"), Integer)
    dsTempQuotation = New DataSet

    Try

        WSProxy.TypeName = "PriceQuote"
        WSProxy.WSDL = dt.Rows(i).Item("WSDL").ToString
        WSProxy.MethodName = dt.Rows(i).Item("WebMethod").ToString
        WSProxy.AddParameter(ProductCode)

        dsTempQuotation = CType(WSProxy.InvokeCall, DataSet)
```

*Figure 6.13: Part of the codes of Web Method RequestForQuotation*

The DynamicWebServicesProxyLib is a class that uses System.Reflection namespace. When the object WSProxy is instantiated, a temporary dynamic assembly is built based on the property WSDL. VBCodeProvider is used to compile the assembly at run-time. VBCodeProvider is a built-in .NET class that provides access to instances of the Visual Basic code generator and code compiler. CreateGenerator() method in VBCodeProvider is invoked to get an object that implements the ICodeGenerator interface. The ICodeGenerator has a number of methods that generate source codes. ICodeGenerator.GenerateCodeFromNamespace method is used to generate code based on the ProxyCodeNamespace. The generated codes are output to the specified text writer, strWriter, as shown in Figure 6.14.

```

'Code Generation using VBCodeProvider
Dim CodeProvider As New VBCodeProvider
Dim ICodeGen As ICodeGenerator = CodeProvider.CreateGenerator
Dim strBuilder As New StringBuilder
Dim strWriter As New StringWriter(strBuilder)
ICodeGen.GenerateCodeFromNamespace(ProxyCodeNamespace, strWriter, _
                                   Nothing)
proxySource = strBuilder.ToString
strWriter.Close()

```

*Figure 6.14: Generate Code From Namespace*

After the codes have been generated, they need to be compiled into an assembly. VBCodeProvider has a method CreateCompiler() that gets an instance of the Visual Basic code compiler which implements ICodeCompiler interface. Figure 6.15 shows the CompileAssemblyFromSource method in the ICodeCompiler is called to compile an assembly from the specified string containing source code, using the specified compiler settings. proxySource is a string variable that contains the source codes that have been generated using ICodeGenerator.

```

Dim compiledAssembly As System.Reflection.Assembly
Dim ICodeComp As ICodeCompiler = CodeProvider.CreateCompiler
Dim resultCompile As CompilerResults
resultCompile = ICodeComp.CompileAssemblyFromSource( _
    paramCompiler, proxySource)
compiledAssembly = resultCompile.CompiledAssembly

```

*Figure 6.15: Compile the Codes to Assembly*

In order to invoke methods in the compiled assembly, an instance or object needs to be created. Activator class is a .NET built-in class that contains methods to create types of objects locally or remotely, or obtain references to existing remote objects. The CreateInstance() method in the Activator class is used to create object of the assembly. The return result of the method is assigned to the proxyInstance variable as depicted as Figure 6.16.



```

Dim assemblyType As Type
assemblyType = compiledAssembly.GetType( _
    "WSEP.Tools.WebServices.DynamicProxy." & objTypeName)
proxyInstance = Activator.CreateInstance(assemblyType)

```

*Figure 6.16: Create Instance from Assembly using Activator Class*

The methods of the instance created can be invoked using the built-in System.Reflection.MemberInfo Class. This class is used to discover the attributes of a member and provide access to member metadata. The proxyInstance is the object which has been instantiated from the compiled assembly using Activator.CreateInstance(). The proxyInstance.GetType().GetMethod(MethodName) retrieves the specific public method information based on the MethodName. MethodName is the public property of class DynamicWebServicesProxyLib. As shown in Figure 6.17, the MethodName is assigned with the WebMethod retrieved from the table VendorWS in the database.

The Invoke() method in the MethodInfo class is used to invoke the method represented by the current instance, using the specified parameters. In this context, when the Invoke() method is called, the web method, as shown in figure 6.13, of the supplier Web Services is invoked. The return result is the dataset of the inventory information.

```

Public Function InvokeCall() As Object

    Dim mi As MethodInfo = proxyInstance.GetType().GetMethod(MethodName)
    Dim result As Object
    result = mi.Invoke(proxyInstance, _
        CType(m_methodParams.ToArray(GetType(Object)), Object{}))
    Return result

End Function

```

*Figure 6.17: Public Function InvokeCall() in Class DynamicWebServicesProxyLib*

## 6.4 Conclusion

The software and hardware development environment are identified. The implementations of all the tiers in the WSEP System are explain in details. Besides, the implementation of the dynamic aspect of the system using Microsoft .NET Reflection is discussed.

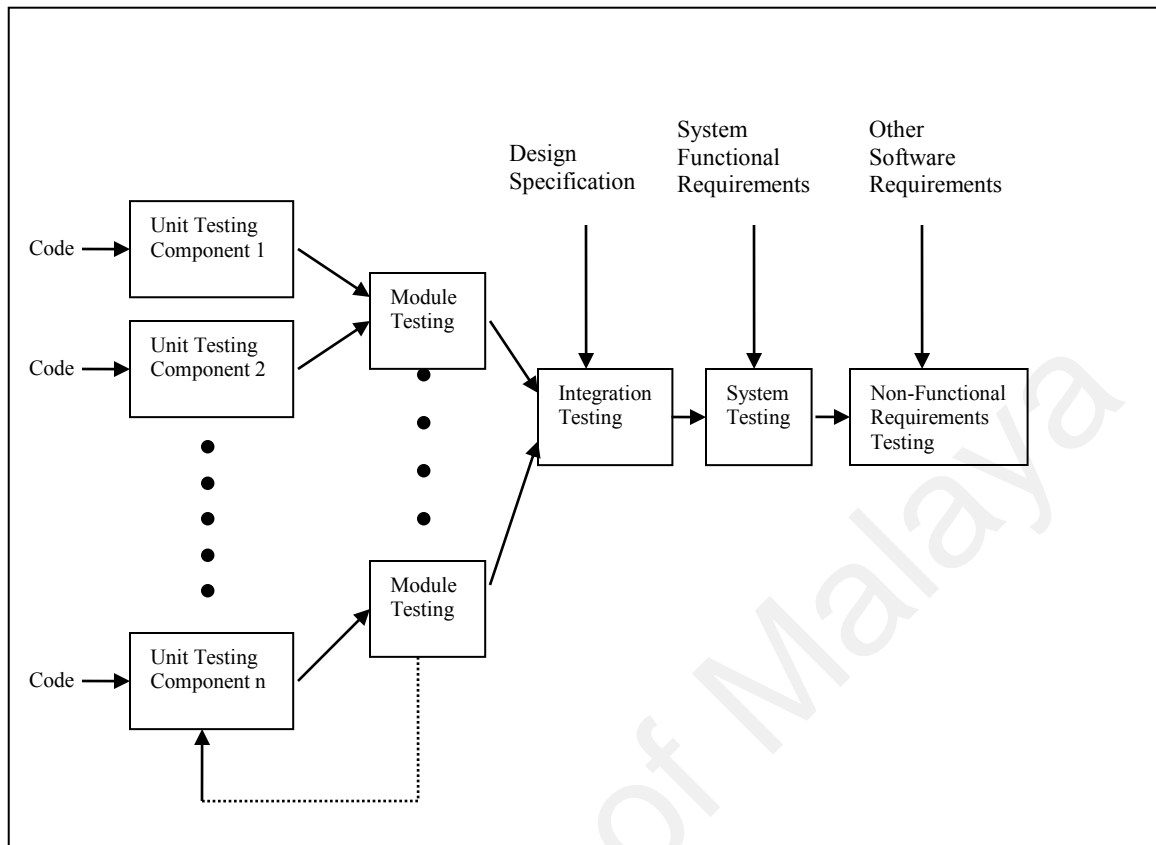
University of Malaya

## CHAPTER 7: SYSTEM TESTING

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results (Hetzel, 1988). There is a plethora of testing methods and testing techniques; each type of testing validates different aspects of the software. Classified by purpose, software testing can be divided into correctness testing, performance testing, reliability testing and security testing. Categorized by life-cycle phase, software testing can be classified into requirements phase testing, design phase testing, program phase testing, evaluating test results, installation phase testing, acceptance testing and maintenance testing. By scope, software testing can be categorized into unit testing, component testing, integration testing, and system testing. Software metrics provide a quantitative basis for the development and validation of models of the software development process. Metrics can be used to improve software quality.

### 7.1 Testing Approach

The types of testing that are to be executed depends on the stage in the development and testing. The process of WSEP System testing follows a pattern that can be shown in the Figure 7.1. The first stage in the testing process involves unit testing. The multiple boxes of Unit Testing in Figure 7.1 show that there are multiple units to test. After the Unit Testing, there is the module testing stage, which involves multiple units to be tested as single module. If any of these modules are defective, it is necessary to return to unit testing. However, this return path is optional, and is performed only if the module testing failed. The dotted arrow in Figure 7.1 indicates this optional path. When all of these are done, the Integration Testing begins by combining all the individual modules together. Finally the whole system is tested against the system functional requirements. Other non-functional requirements are also tested.



*Figure 7.1 Testing Process*

### 7.1.1 Unit Testing

In this stage, individual components are tested to ensure that they operate correctly. Each component is tested independently without other system components. The aim of unit testing is to isolate each part of the program and show that the individual parts are correct. For this project, individual Web Services method is considered as a single unit. Each Web Services method is tested against defects. There are several approaches to unit testing. Black box testing and white box testing have been used for this testing purpose.

#### 7.1.1.1 Black Box testing

Black box testing relies on the specification of the component which is being tested to derive test cases. It only concerns about what inputs are put in, and what outputs are generated as a result of the test. It is a functional testing procedure to validate if the unit meets its specifications. The unit is a 'black box' whose behavior can be determined by studying its inputs and outputs.

Figure 7.2 shows the test input page for GetVendorProduct unit of the Web Services. The input is the VendorID which is the parameter of the Web Services method. The return result of the method is in XML format, which may look like Figure 7.3. The output of the unit is then studied. If the output shows the correct result, the test is successful and vice versa.

The screenshot shows a web interface titled "WSeProcurement" in a dark blue header. Below the header, there is a link "Click [here](#) for a complete list of operations." followed by a horizontal line. The main section is titled "GetVendorProduct" in bold. Underneath, the word "Test" is followed by the instruction: "To test the operation using the HTTP POST protocol, click the 'Invoke' button." Below this instruction is a table with two columns: "Parameter" and "Value". The first row of the table has "VendorID:" in the "Parameter" column and a text input field containing "1" in the "Value" column. To the right of the table is a button labeled "Invoke".

Parameter	Value
VendorID:	<input type="text" value="1"/>

Invoke

*Figure 7.2 Black Box Testing of a Web Service*

```

<?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="http://tempuri.org/WSEProcurement/WSeProcument">
- <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
- <xs:element name="NewDataSet" msdata:IsDataSet="true">
- <xs:complexType>
- <xs:choice maxOccurs="unbounded">
- <xs:element name="Table">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="VendorProductID" type="xs:int" minOccurs="0" />
  <xs:element name="ProductID" type="xs:int" minOccurs="0" />
  <xs:element name="ProductCode" type="xs:string" minOccurs="0" />
  <xs:element name="ProductName" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>
- <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
  xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
- <NewDataSet xmlns="">
- <Table diffgr:id="Table1" msdata:rowOrder="0">
  <VendorProductID>7</VendorProductID>
  <ProductID>4</ProductID>
  <ProductCode>F252</ProductCode>
  <ProductName>Computer Table</ProductName>
</Table>
</NewDataSet>
</diffgr:diffgram>
</DataSet>

```

*Figure 7.3 Correct Result of Black Box Testing*

#### 7.1.1.2 White Box Testing

Another complementary approach to black box testing is white box testing. It is a behavioral or structural testing. The testing is done with the knowledge of the code used to execute certain functionality. Figure 7.4 shows the structure of a unit in the Web Services method, GetVendorProduct, is analyzed.

```

<WebMethod()> _
Public Function GetVendorProduct (ByVal VendorID As Integer) As DataSet

    Dim objVendor As New bllVendor

    Try
        Return objVendor.GetVendorProduct (VendorID)
    Catch ex As Exception
        Throw New Exception(ex.Message, ex)
    Finally
        If Not objVendor Is Nothing Then objVendor = Nothing
    End Try
End Function

```

```

Public Function GetVendorProduct (ByVal VendorID As Object) As DataSet

    Dim objVendor As New dalVendor

    Try
        Return objVendor.GetVendorProduct (VendorID)
    Catch ex As Exception

        Throw New Exception(ex.Message, ex)

    Finally

        If Not objVendor Is Nothing Then objVendor = Nothing

    End Try

End Function

```

```

Public Function GetVendorProduct(ByVal VendorID As Object) As DataSet

    Dim strSQL As String = ""           ' Used for the sql SELECT clause
    Dim strWHERE As String = ""         ' Used for the sql WHERE clause

    ' Build SELECT clause
    strSQL = strSQL & "SELECT [VendorProduct].[VendorProductID] "
    strSQL = strSQL & "          , [Product].[ProductID] "
    strSQL = strSQL & "          , [Product].[ProductCode] "
    strSQL = strSQL & "          , [Product].[ProductName] "

    strSQL = strSQL & " FROM Product "
    strSQL = strSQL & " INNER JOIN VendorProduct "
    strSQL = strSQL & " ON Product.ProductID = VendorProduct.ProductID "
    strSQL = strSQL & " WHERE VendorProduct.VendorID = "
    strSQL = strSQL & VendorID.ToString()

    ' Execute the SQL and return the data
    Try
        Return SqlHelper.ExecuteDataset(Globals.ConnectionString, _
            CommandType.Text, strSQL)

    Catch ex As Exception
        Throw New Exception(ex.Message, ex)
    Finally

    End Try

End Function

```

#### 7.4 White Box Testing: Examining and analyzing the code structure

##### 7.1.2 Module Testing

Once all the units have been tested, it is necessary to ensure that these units work together as a single module. A module encapsulates all the related components so it can be tested without other modules. Figure 7.5 shows vendor module that consists of User Interface, Web Services, blVendor and dalVendor components. Module testing is done against all the functionalities for the vendor module



Maintenance

Vendors

Products

Vendor Web Services

Procurement

Request For Quote

Purchase Order

Common

Logout

## Vendor Listing

New

Action	Vendor ID	Vendor Code	Vendor Name
[ Edit   Delete   Products ]	1	V023	Bend New Inc
[ Edit   Delete   Products ]	2	V3453	Alain Company
[ Edit   Delete   Products ]	3	V033	Bend New Inc

1

```

#Region " Vendor "
    <WebMethod()> _
    Public Function GetVendor(ByVal VendorID As Integer, _
        ByVal VendorCode As String, _
        ByVal VendorName As String) As DataSet...

    <WebMethod()> _
    Public Function AddVendor(ByRef VendorID As Integer, _
        ByVal VendorCode As String, _
        ByVal VendorName As String) As Boolean...

    <WebMethod()> _
    Public Function UpdateVendor(ByVal VendorID As Integer, _
        ByVal VendorCode As String, _
        ByVal VendorName As String) As Boolean...

    <WebMethod()> _
    Public Function DeleteVendor(ByVal VendorID As Integer) As Boolean...

    <WebMethod()> _
    Public Function GetVendorProduct(ByVal VendorID As Integer) As DataSet...

    <WebMethod()> _
    Public Function AddVendorProduct(ByVal VendorID As Integer, _
        ByVal Products As ArrayList) As Boolean...

    <WebMethod()> _
    Public Function DeleteVendorProductByKey(ByVal VendorProductID As Integer) As Boolean...
#End Region

```

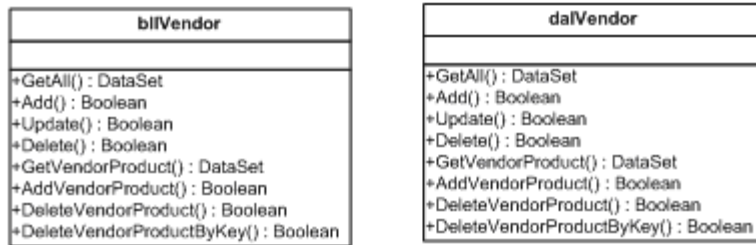


Figure 7.5: Units in a Vendor Module

### 7.1.3 Integration Testing

In integration testing, individual modules are combined together to test as a group. This testing is concerned with identifying errors which result from unexpected interactions between modules. There are two most common strategies used for integration testing:

- Top-down approach that requires the highest-level modules be tested and integrated first. This allows high-level logic and data flow to be tested early in the process and it tends to minimize the needs for drivers.
- Bottom-up approach requires the lowest-level units tested and integrated first. By using this approach, units or modules are tested early in the development process and the need for stubs is minimized.

Bottom-up approach is used for integration testing. There are no problems and issues found during the integration testing since all the modules and components are built using .NET 1.1 framework.

### 7.1.4 System Testing

System testing involves testing of a complete application environment that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware to evaluate the system's compliance

with its specified requirements. System testing falls within the scope of black box testing, and thus, should require no knowledge of the inner design of the code. During this testing, the whole system is tested against functional requirements and system requirements that have been specified during system analysis phase. Besides, there is also the need to test for user interface, performance, security and reliability for the system.

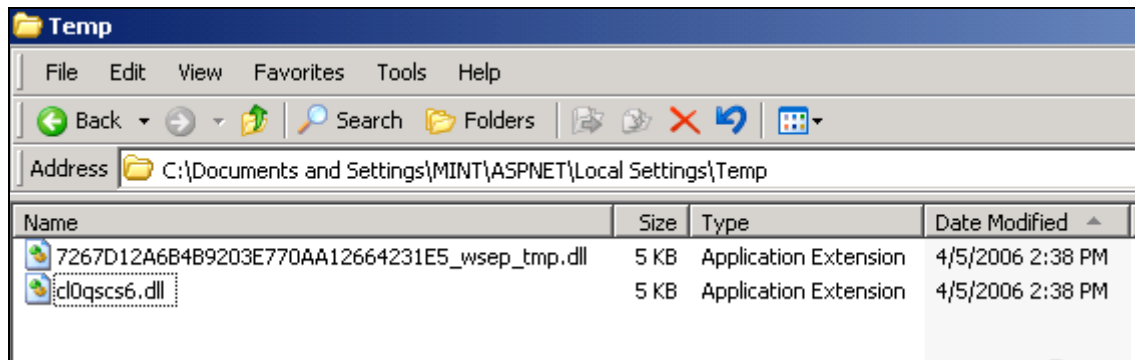
## **7.2 Non-Functional Requirements Testing**

- **Security Testing**

The system is protected from unauthorised access. This is done by providing user authentication and using secure channel like SSL to encrypt the data transmits. The system should provide a means to enter user names and passwords. All the web pages are protected. If an unauthorized user tries to access any pages, the system will redirect the user to the login page to enter username and password.

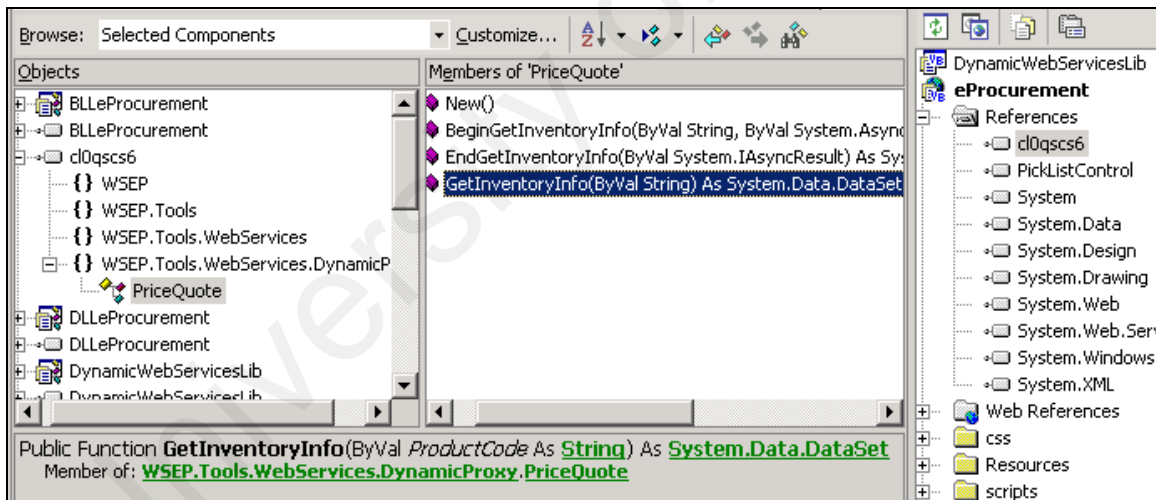
- **Dynamic Requirement Testing**

The system should provide dynamic and real time operations to the user. The Supplier Web Services should be invoked at run-time instead of design time. In order to check if the assemblies are created during run-time, a break point is set in the DynamicWebServicesProxyLib class where the assembly is compiled and created. By tracing into the code, the temporary path of dynamic assembly is identified, as shown in Figure 7.6. There are a few assemblies created in the folder during the execution of the codes.



*Figure 7.6 Assemblies that are Created Dynamically*

In order to examine the assembly created, Visual Studio .NET Object Browser is used. The assembly is referenced and imported. As shown in Figure 7.7, the assembly has the namespace `WSEP.Tools.WebServices.DynamicProxy` and has a class `PriceQuote` with the function `GetInventoryInfo()`. This assembly acts as the dynamic proxy that will bind to Supplier Web Services.



*Figure 7.7 Visual Studio .NET Object Browser Is Used to Examine the Dynamic Assembly*

- **Performance Testing**

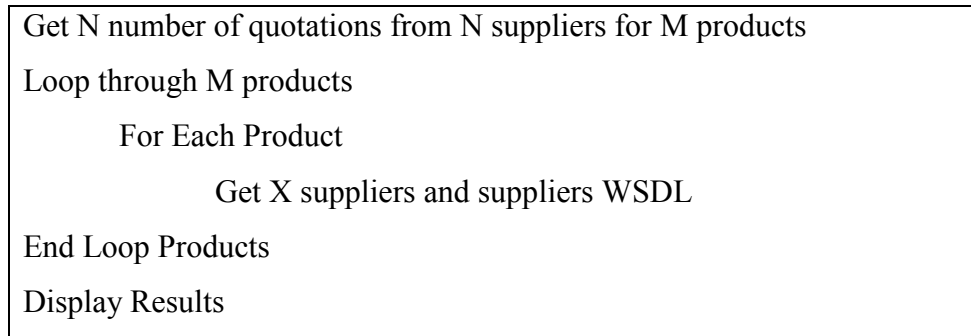
In software engineering, performance testing is the testing that is performed to determine how fast some aspect of a system performs under a particular workload. Performance testing is to demonstrate that the system meets performance criteria. It

involves obtaining data concerning how well the system executes the functions they were designed for.

During the testing process, system attributes such as CPU usage, memory usage, SQL Server response time are captured to gauge the performance of the system. In order to do this, a Windows Form test program written in .NET is built. The performance test screen is shown in Figure 7.8. All the tests are conducted using the same algorithm as in Table 7.1. The test involves looping of M records of products multiply with X records of suppliers for each product.

VendorID	VendorCode	VendorName	WSDL	ProductCode	ProductName
1098	VC1098	VN1098	http://192.168.1.10/Vendor1WS/	PC1010	PN1010
1099	VC1099	VN1099	http://192.168.1.10/Vendor1WS/	PC1010	PN1010
1100	VC1100	VN1100	http://192.168.1.10/Vendor1WS/	PC1010	PN1010
1101	VC1101	VN1101	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1102	VC1102	VN1102	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1103	VC1103	VN1103	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1104	VC1104	VN1104	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1105	VC1105	VN1105	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1106	VC1106	VN1106	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1107	VC1107	VN1107	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1108	VC1108	VN1108	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1109	VC1109	VN1109	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1110	VC1110	VN1110	http://192.168.1.10/Vendor1WS/	PC1011	PN1011
1111	VC1111	VN1111	http://192.168.1.10/Vendor1WS/	PC1012	PN1012
1112	VC1112	VN1112	http://192.168.1.10/Vendor1WS/	PC1012	PN1012
1113	VC1113	VN1113	http://192.168.1.10/Vendor1WS/	PC1012	PN1012
1114	VC1114	VN1114	http://192.168.1.10/Vendor1WS/	PC1012	PN1012
1115	VC1115	VN1115	http://192.168.1.10/Vendor1WS/	PC1012	PN1012

Figure 7.8: Windows Forms Performance Test Screen



*Figure 7.9: Algorithm for Performance Testing of Request For Quotation Transaction*

The results of the tests are shown in Table 7.1, Figure 7.10 and Figure 7.11. The usage of CPU, memory and SQL server response time are considered normal.

*Table 7.1: Test Result of Performance*

Number of Products	Total Records Returned	CPU Usage (%)	Memory Usage (%)	SQL Duration (milliseconds)
1	10	2.0	3.0	60
10	100	4.0	4.0	60
20	200	5.0	7.0	60
50	500	8.0	11.0	200
100	1000	19.0	19.0	240
200	2000	27.0	29.0	303
300	3000	36.0	37.0	310
400	4000	43.0	44.0	370
500	5000	47.0	49.0	430

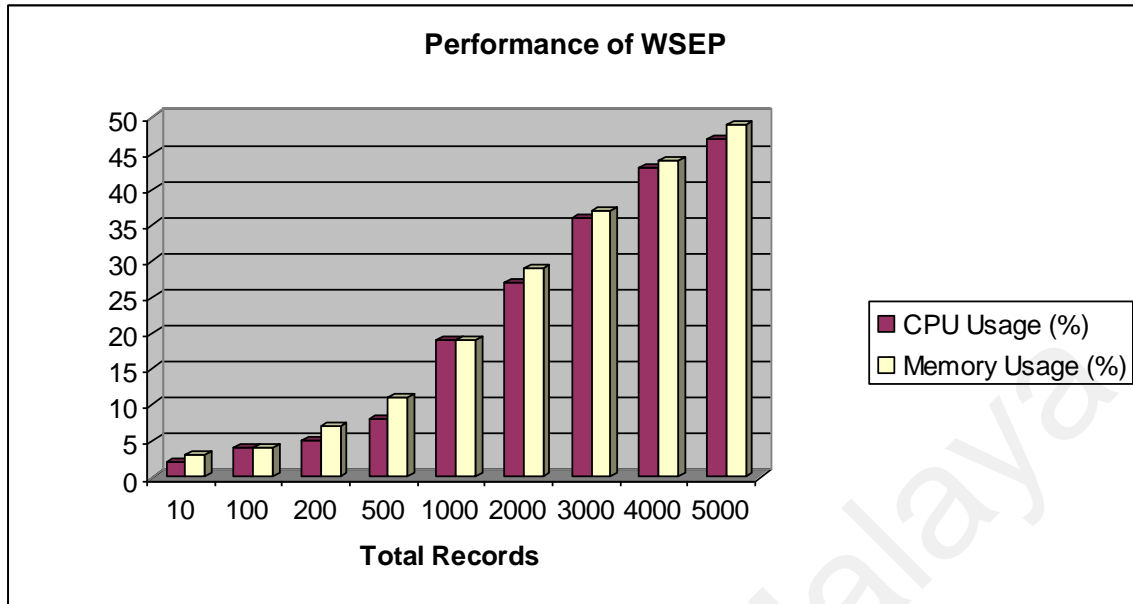


Figure 7.10: Bar Chart for Test Result of CPU usage and Memory usage

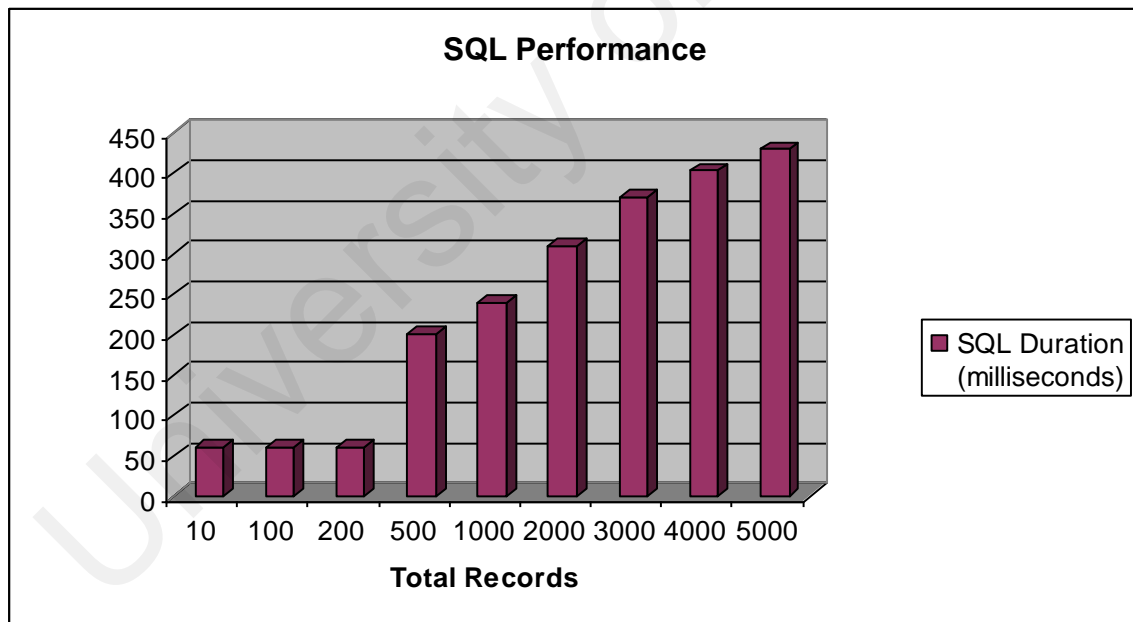
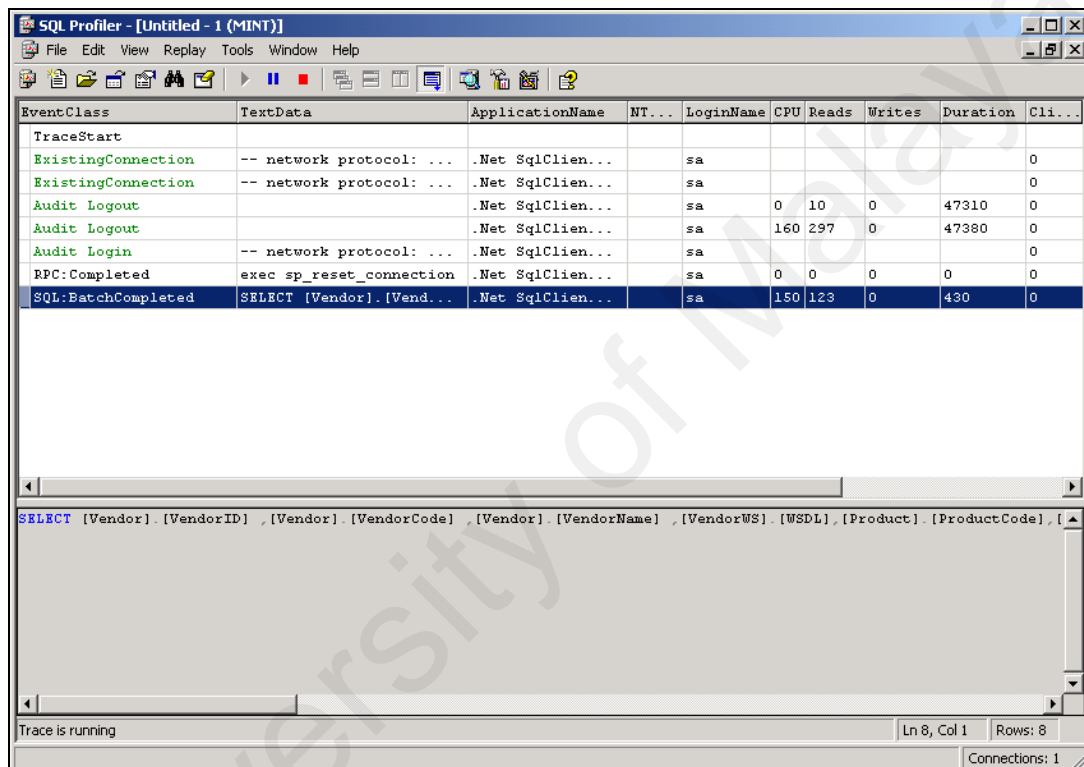


Figure 7.11: Bar Chart for Test Result of SQL Response Time

Figure 7.12 shows the SQL Profiler that is used to get the SQL response time during the testing process. The column Duration shows the execution time in millisecond for the SQL query. Figure 7.13 show the operating system resources usage during the testing process. The results of performance testing show no significant degradation in performance when the number of query records increase.



EventClass	TextData	ApplicationName	NT...	LoginName	CPU	Reads	Writes	Duration	Cli...
TraceStart									
ExistingConnection	-- network protocol: ...	.Net SqlClie...		sa					0
ExistingConnection	-- network protocol: ...	.Net SqlClie...		sa					0
Audit Logout		.Net SqlClie...		sa	0	10	0	47310	0
Audit Logout		.Net SqlClie...		sa	160	297	0	47380	0
Audit Login	-- network protocol: ...	.Net SqlClie...		sa					0
RPC:Completed	exec sp_reset_connection	.Net SqlClie...		sa	0	0	0	0	0
SQL:BatchCompleted	SELECT [Vendor].[Vend...	.Net SqlClie...		sa	150	123	0	430	0

```

SELECT [Vendor].[VendorID] , [Vendor].[VendorCode] , [Vendor].[VendorName] , [VendorWS].[WSDL] , [Product].[ProductCode] , [

```

Trace is running      Ln 8, Col 1      Rows: 8      Connections: 1

Figure 7.12: SQL Profiler for SQL Response Time Testing



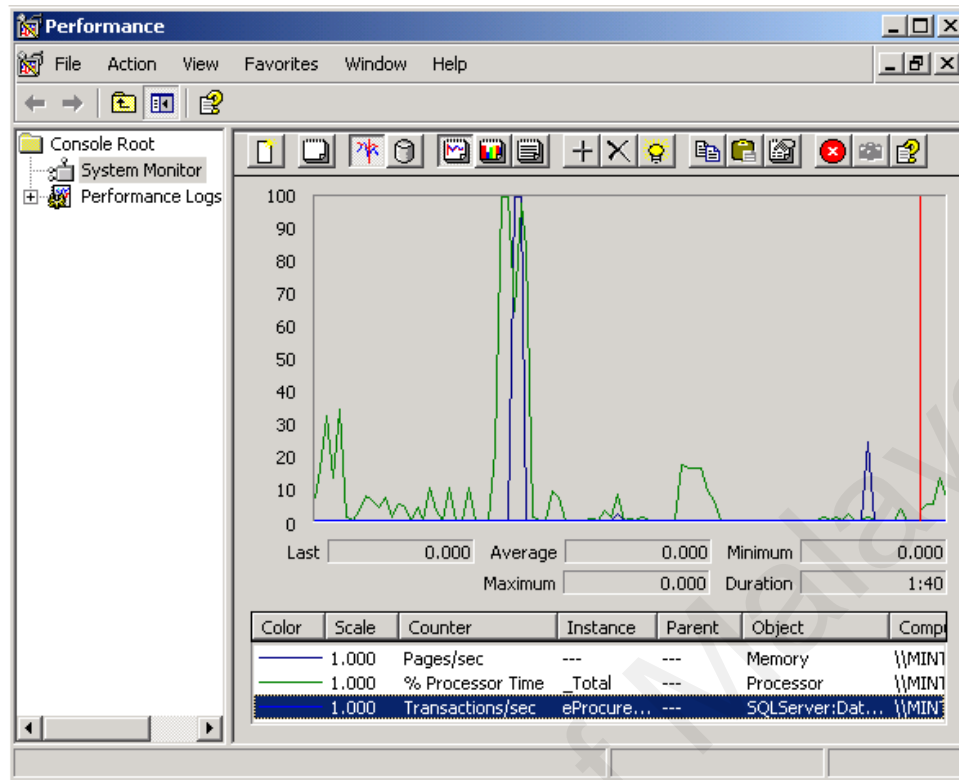


Figure 7.13: Windows Performance Utility Tool for Memory and CPU Usage Testing

### 7.3 WSEP Metrics

Software metrics can be defined as activities concerned with measurement in software engineering (Fenton and Neil, 1999). Software metrics provide feedback about the external quality aspects of software system such as maintainability, reliability and reusability and provide a ways to estimate the effort needed for testing. They measure different aspects of software complexity and therefore play an important role in analyzing and improving software quality (Briand, 1999).

In recent years many researchers and practitioners have proposed a number of code metrics for object oriented software. Traditional metrics for measuring software such as Lines of Code (LOC) have been found inadequate for analysis of object-oriented software (Fenton and Neil, 1999). The complexity of an object-oriented design may be quantified by measures that assess its internal quality. Many metrics have been proposed to statically evaluate the quality of a software design. Two such measures are coupling and cohesion (Mitchell, 2002).

Figure 7.14 depicts how different measures of software complexity can characterize the overall quality of a software product. Internal Quality measures are those which can be performed in terms of the software product itself. They can be evaluated by measuring the Coupling and Cohesion of the software design.

Internal Quality measures have no practical meaning within themselves. To give meaning to these measures, they have to be characterized in terms of the External Quality of a software product (Mitchell, 2002). External Quality measures are evaluated with respect to how a product relates to its environment. The maintainability, reliability and reusability are examples of External Quality that will be measured in this project. These measures are considered to be inherently meaningful.

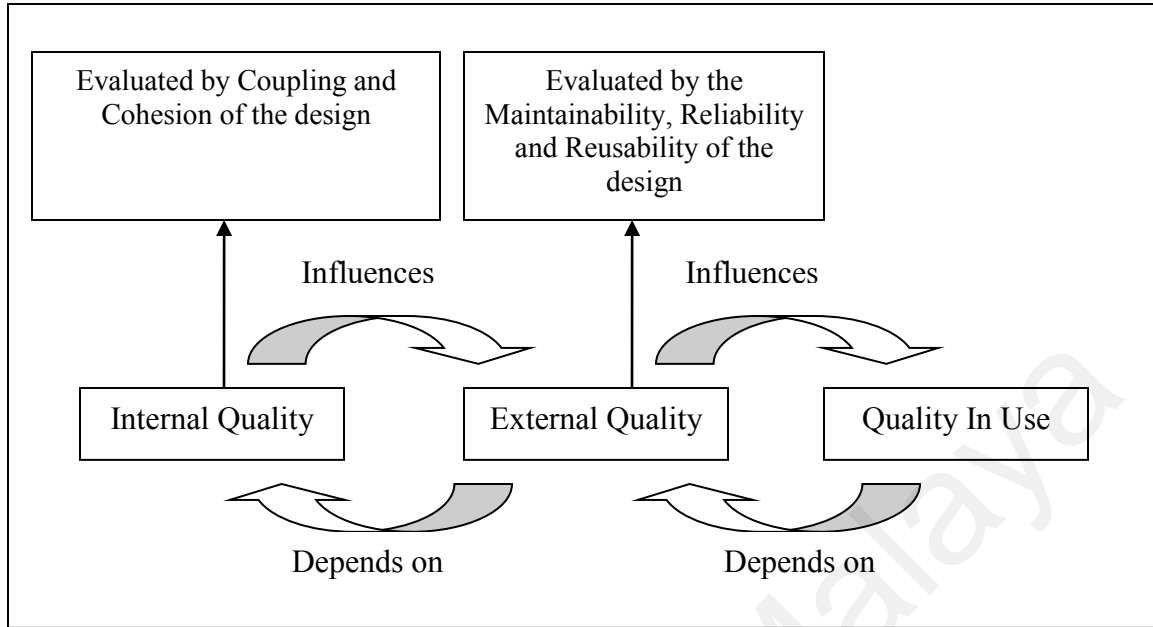


Figure 7.14: Software Quality Model (Mitchell, 2002)

### 7.3.1 Coupling

Coupling is defined as “two classes are coupled when methods declared in one class use methods or instance variables of the other class” (Chidamber and Kemerer, 1994). In ontological terms, "two objects are coupled if and only if at least one of them acts upon the other, X is said to act upon Y if the history of Y is affected by X, where history is defined as the chronologically ordered states that a thing traverses in time" (Vessey and Weber, 1984).

Let  $X = \langle x, p(x) \rangle$  and  $Y = \langle y, p(y) \rangle$  be two objects.

$$p(x) = \{ M_X \} \cup \{ I_X \}$$

$$p(y) = \{ M_Y \} \cup \{ I_Y \}$$

where  $\{ M_i \}$  is the set of methods and  $\{ I_i \}$  is the set of instance variables of object  $i$ .

Using the above definition of coupling, any action by  $\{ M_X \}$  on  $\{ M_Y \}$  or  $\{ I_Y \}$  constitutes coupling, as does any action by  $\{ M_Y \}$  on  $\{ M_X \}$  or  $\{ I_X \}$ . When  $M_X$  calls  $M_Y$ ,  $M_X$  alters the history of the usage of  $M_Y$ ; similarly when  $M_X$  uses  $I_Y$ , it alters the access and usage history of  $I_Y$ . Therefore, any evidence of a method of one object using methods

or instance variables of another object constitutes coupling. Since objects of the same class have the same properties, two classes are coupled when methods declared in one class use methods or instance variables of the other class (Shyam and Chris, 1993).

A class is coupled to another class if it uses the member method and/or instance variables of the class. A good software design should minimize coupling between objects. This will promote reusability, maintainability and testability of a class (Chidamber and Kemerer, 1994). A high level of coupling is undesirable as it prevents reuse and is detrimental to modular design. The more independent a class is, the easier it is to reuse it in another application. The larger the number of couples, the higher the sensitivity to changes in other parts of the design; maintenance is therefore more difficult (Shyam and Chris, 1993).

Coupling Between Objects (CBO) is used in this project as a metric to measure coupling of the WSEP system.

### **Metric 1: Coupling Between Objects (CBO)**

CBO is defined as the count of the number of other classes to which it is coupled (Chidamber and Kemerer, 1994). Two classes are coupled when methods declared in one class use methods or instance variables defined by the other class. Multiple accesses to the same class are counted as one access. Only method calls and variable references are counted. Other types of reference, such as use of constants, handling of events, use of user-defined types and object instantiations are ignored.

#### **7.3.2 Cohesion**

Bunge defines *similarity*  $\sigma()$  of two things to be the intersection of the sets of properties of the two things (Bunge, 1997):

$$\sigma(X,Y) = p(x) \cap p(y)$$

Based on this principle of defining similarity, the degree of similarity of the methods within the object can be defined to be the intersection of the sets of instance variables that

are used by the methods. It should be clearly understood that instance variables are not properties of methods, but it is consistent with the notion that methods of an object are intimately connected to its instance variables (Shyam and Chris, 1993):

$$\sigma (M_1, M_2) = \{ I_1 \} \cap \{ I_2 \}$$

where  $\sigma (M_1, M_2)$  = degree of similarity of methods  $M_1$  and  $M_2$

and  $\{ I_i \}$  = set of instance variables used by method  $M_i$

Example: Let  $\{I_1\} = \{a,b,c,d,e\}$  and  $\{I_2\} = \{a,b,e\}$ .  $\{I_1\} \cap \{I_2\}$  is non-empty, and  $\sigma (M_1, M_2) = \{a,b,e\}$ .

The degree of similarity of methods in an object class are related both to the conventional notion of cohesion in software engineering as well as encapsulation. The degree of similarity of methods can be viewed as a major aspect of object class cohesiveness (Shyam and Chris, 1993). The cohesion of a class is the degree to which its methods are related to each other. It is determined by examining the pattern of state variable accesses within the set of methods. If all the methods access all the state variables within the class then they have high cohesion; if they access disjoint sets of variables then the cohesion is low. If a class exhibits low method cohesion it indicates that the design of the class has probably been partitioned incorrectly. This is not desirable. If a class performs several unrelated functions, it should be split up into more classes with individually higher cohesion. On the other hand, a high value of cohesion implies that the class is well-designed. A cohesive class will tend to provide a high degree of encapsulation, whereas a lack of cohesion decreases encapsulation and increases complexity.

Cohesion metrics measure how well the methods of a class are related to each other. For measuring cohesiveness of WSEP system, the improved Lack of Cohesion Of Methods (LCOM) (Hitz & Montazeri, 1995) is used.

## **Metric 2: Lack of Cohesion of Methods (LCOM) By Hitz and Montazeri**

Lack of Cohesion in Methods (LCOM) is defined as “the number of pairs of methods in a class, having no common attributes, minus the number of pairs of methods sharing at least one attribute” (Chidamber and Kemerer, 1994). The metric is set to zero when the value is negative.

Li and Henry redefine LCOM as the number of disjoint sets of methods accessing similar instance variables (Li and Henry., 1993). Hitz and Montazeri restate Li’s definition of LCOM based on graph theory. LCOM is defined as the number of connected components of a graph (Hitz and Montazeri, 1995). A connected component is a set of related methods and class-level variables. There should be only one such a component in each class. If there are 2 or more components, the class should be split into smaller classes. In other words, if the LCOM value is equal to or higher than 2, the system has low cohesion which is undesirable.

The assumption behind the cohesion metrics for the WSEP system is that methods are related if they work on the same class level variables; and methods are unrelated if they work on different variables. Constructors and destructors are ignored in the count of LCOM. This is because they frequently initiate or clear all variables in the class and making all methods connected through these variables, which will increase cohesion artificially.

### **7.3.3 Analysis of the Result**

Table 7.2 shows the results in details for each of the modules being analyzed for CBO and LCOM. There are all together 23 classes with around 4000 lines of code are analyzed. A bar chart result is shown Figure 7.15.

Table 7.2: Results of Each Module Being Analyzed

Module	Authentication	Maintenance	E-Procurement Transaction WS
Classes	3	17	3
Lines of Code	160	2575	1267
CBO	3.4	3.5	3.85
LCOM	1.28	1.25	1.36
Quality	Good	Good	Medium

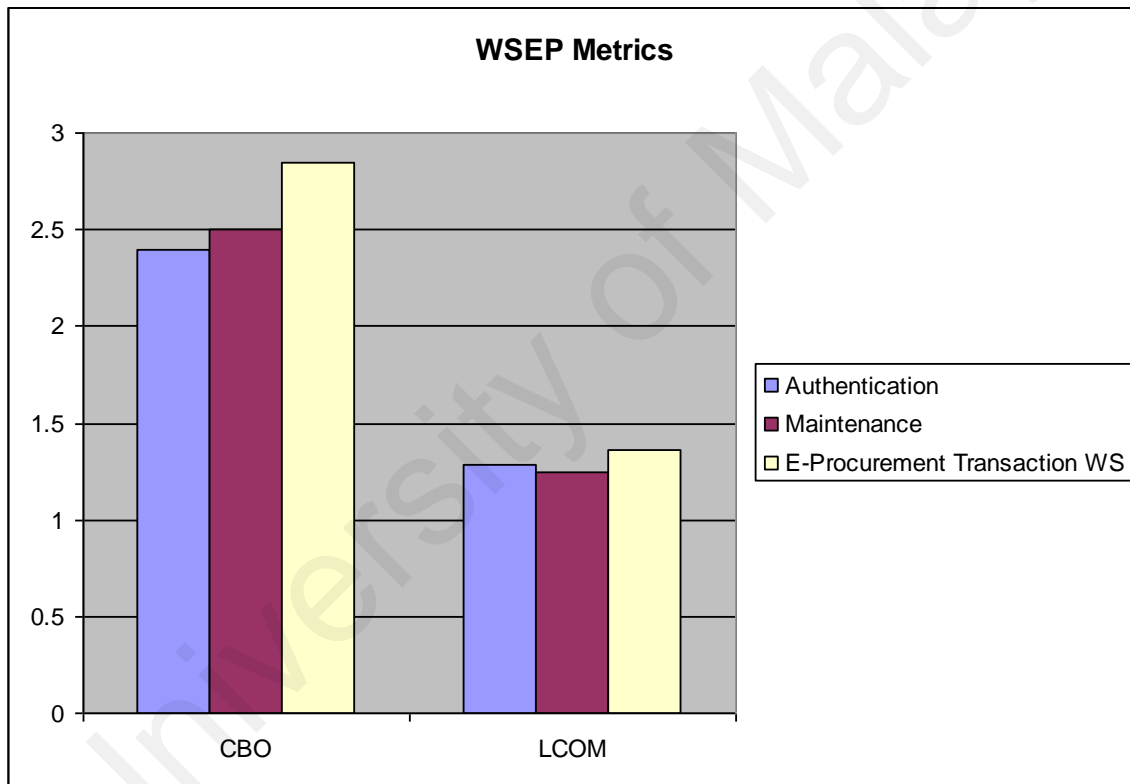


Figure 7.15: Bar Char for the WSEP Metrics Result

According to Houari, CBO value which is higher than 14 is considered too high (Houari, Robert, Thierry, 2005). The WSEP system has CBO average value of 3.58, which is considered low. This means that the WSEP system has low coupling where the objects in the system have high degree of independence and are less coupled from each

other. The higher the degree of object independence, the more likely it is that objects will be suitable for reuse within the same applications and within other applications.

Besides, a higher degree of coupling between objects is likely to complicate application maintenance because object interconnections and interactions are more complex. Uncoupled objects are easier to augment and maintain than those with a high degree of dependencies.

The WSEP system has the average LCOM value of 1.28 which is closed to 1. In other words, the objects in system have high cohesion. High cohesion is desirable since it promotes encapsulation. In object-oriented programming, it is good to assign a single logical task to a class. This keeps cohesion high and maximizes reusability, while complexity is kept manageable.

Based on the result of CBO and LCOM, the WSEP system is said to have low coupling and high cohesion. In conclusion, the low level of coupling and high level of cohesion promote reuse, increase the maintainability and reliability of the system.

## **7.4 Conclusion**

Software testing is to assure the system performs the functionality for which it was built. A few testing methods which include unit testing, integration testing, system testing were carried out to ensure the system meet its requirements. Besides, three of the non-functional requirements were tested. WESP has passed all these tests. In addition, two of the metrics techniques were applied for software quality measurement.



## **CHAPTER 8: CONCLUSION**

### **8.1 Problems Encountered**

There are a few problems and issues encountered during the development of WSEP system. This section will discuss the problems and its solutions for each of them.

- **Difficulties in Dynamic Binding of Web Services**

Although there are a lot of documentations regarding creating Web Services applications, none of them explain how to dynamically bind the Web Services. All of the documentations in Microsoft Developer Network (MSDN) explain the usage of Web Services by static binding with the proxy using Visual Studio .NET 2003.

In order to find some solutions to this problem, the Web Sites of newsletters were consulted and questions were posted in the forums in addition to Microsoft's MSDN web site. However, the information is inadequate and is not readily available.

- **Difficulties in Getting Business Domain Knowledge**

Lacking of exposure in E-Procurement business process had impeded the progress of analysis phase. However, by reading books and doing research in this field, the process flow of E-Procurement was identified.

- **Problems with Web Forms**

There were some problems related to the use of ASP.NET Web Forms. The .NET Web Services application is stateless programming model. The consequence is that each incoming request is handled independently of the previous ones. The only state maintained between requests is anything that saved or persisted in a data store.

## 8.2 System Strengths

The strengths of WSEP lie in the use of technology that overcomes many of the weaknesses of the traditional system that uses EDI. The following are some of the strengths:

- **Open Standard**

WSEP system is implemented using Web Services. Web services are software components that communicate using pervasive and standards-based Web technologies, including HTTP and XML-based messaging. Web Services are designed to be accessed by other applications. Since they are based on open standards such as HTTP and XML-based protocols including SOAP and WSDL, Web services are hardware, programming language, and operating system independent. This means that system written in different programming languages and running on different platforms can seamlessly exchange data over intranets or the Internet using Web Services.

- **Provide Real Time Information**

Web Services provide an easy way for integration. Web Services have evolved as a practical and cost-effective solution for uniting information distributed among applications. By querying through Web Services, WSEP system can retrieve information from the supplier legacy systems. Any updates or changes in the supplier systems will be made available immediately.

- **N-Tier Architecture is very flexible**

The n-tier architecture of WSEP made it very easy to extend and add on new modules. The system is modular and allows the addition of new modules without changing the overall architecture of the whole system.

- **Reliability**

WSEP system uses .NET framework and Microsoft SQL Server that are built for

enterprise level of applications. The system relies on a proven technology that has been developed and tested by millions of users worldwide.

### **8.3 System Constraints and Future Enhancement**

- **System Constraints**

The WSEP system is implemented using local server with a private registry that keeps data on the database. As there is no public UDDI, it is not possible for the information to be published over the Internet as a whole. There is no real-world environment for testing the system. However, theoretically, the same system can work over the Internet once a permanent Internet Service Provider (ISP) link can be established, and a Web Server configured for this purpose is set up.

- **Future Enhancements**

A number of potential extensions and enhancements of the project have been identified. These are discussed in the following:

#### **Enhance User Interface**

Since the WSEP system is implemented using Web Services, the user interface can be implemented using other user interface components, such as mobile devices and Personal Digital Assistants (PDA). All these components work fine with Web Services. However, this will require further development to design components for downloading web pages into mobile devices and PDA.

#### **Performance Enhancement**

By configuring IIS, we can allow ASP.NET handles incoming requests by servicing them with a pool of threads. Usually, when a request comes in, a thread from the pool that is idle will pop in to service the incoming request. However, there is a limit of threads that can be created to handle a large number of requests. These factors limit the performance of Web Service requests.

The current WSEP system is using synchronous Web Services. We can further improve the performance of the WSEP system by using asynchronous Web Services calls with callback functions and multithreading. There can be a performance benefit of up to a few times the number of service requests processed.

### **Web Services Security**

Security has always been a main problem of implementing Web Services. SSL has been used in this project to secure the communication between the channels. There is a need to apply more security measures like authentication of the clients who call the Web Services to prevent unauthorized access of the services.

## **8.4 Knowledge and Experience Gained**

This project has been a very rewarding experience. There are many knowledge and experience gained during the development of this project which includes:

- **.NET Framework**

.NET Framework is powerful platform to build enterprise application. There are many built-in classes and library in this Framework, and many lessons were learned when using and applying its concepts when developing this system.

The .NET Framework has been built with Web Services in mind. Therefore creating Web Services application with .NET is not difficult. It provides full support for using web standards such as eXtensible Markup Language (XML) and Simple Object Access Protocol (SOAP) in a seamless manner.

- **Visual Studio.NET**

Visual Studio.NET is an easy to use and user friendly IDE. There are step by step tutorials that guide the user of using it. The debugging tool helps a lot in tracing the code

and finding the errors.

- **ASP.NET**

ASP.NET is very much different from ASP. ASP.NET is a unified web development model that includes the services necessary for building enterprise-class web applications with a minimum of coding. It offers many new opportunities to learn about programming web pages that are interactive and dynamic. VB.NET has been used as code behind to develop this project. This enables the ASP.NET application to benefit from the common language runtime, type safety, inheritance and so on.

- **ADO.NET**

ADO.NET is a database access technology that offers several advantages over previous versions of ADO and over other data access components which includes interoperability, performance, maintainability and scalability. The use of the Dataset has been very exciting, as they allow easy retrieval of data and direct binding to DataGrid web control.

- **Web Services**

Web services extend the World Wide Web infrastructure to provide the means for software to connect to other software applications. Web Services combine the best aspects of component-based development and the Web. This project has provided means to learn more about Web Services.

## 8.5 Conclusion

By using the dynamic nature of Web Services, this project shows the design and implementation of an E-Procurement system in the context of Supply Chain Management to provide real time information sharing and dynamic procurement operations. Earlier Supply Chain Management which uses EDI technology to integrate the business processes in the Supply Chain seems to be costly and inflexible. With the advent of Web Services, most of the problems faced by earlier technologies can be overcome. Web Services use standard protocols and data formats such as HTTP, SOAP, and XML to connect to other software applications. This project demonstrates the using of Web Services to provide dynamic procurement operations over the internet.

The development of this project uses Web Services and Microsoft .NET technologies that improve upon EDI technology in terms of easy of use in development, scalability and maintainability. The system is designed using n-tier architecture that separates the responsibilities of the system into multiple tiers. This separation provides encapsulation for the different tiers and components, which results in a more scalable and robust system.

## References

- Alonso, G., et. al., (2003). *Web Services – Concepts, Architectures and Applications*. Springer Verlag, Heidelberg, Germany.
- Amit R. and Zott C., (2001). *Value creation in e-Business*. Strategic Management Journal, Vol: 22, Issue:6-7
- Anne T. M., (March 2001), *Enabling Open, Interoperable, and Smart Web Services: The Need for Shared Context*, at <http://www.w3.org/2001/03/WSWS-popa/paper29>, Sun Microsystems
- Archer N. and Yuan Y., (2000). *Managing business-to-business relationships throughout the e-commerce procurement life cycle*. Internet Research: Electronic Networking Applications and Policy, Vol:10
- Barua A., Konana P., Whinston A.B., and Yin F., (2001). *Driving E-Business Excellence*. Sloan Management Review.
- Briand L.C., (March 1999). *Empirical Investigations of Quality Factors in Object-Oriented Software*, Empirical Studies of Software Engineering, Ottawa, Canada
- Bunge, M., (1977), *Treatise on Basic Philosophy : Ontology I : The Furniture of the World*, Boston: Riedel
- Chartier R., (2005) *Application Architecture: An N-Tier Approach - Part I*, at <http://www.15seconds.com/issue/011023.htm>
- Chase N.,(Jun 2002) *Introduction to ebXML* at <https://www6.software.ibm.com/developerworks/education/x-ebxml/>
- Chidamber L., Shyam A. and Chris K., (June 1994). *A Metrics Suite for Object-Oriented Design*, IEEE Transactions on Software Engineering, pp. 476-492
- Christopher M., (1992) *Logistics and Supply Chain Management – Strategies for Reducing Costs and Improving Services*, Pitman Publishing (London)
- Clabby J., (2002). *Web Services Explained: Solutions and Applications for the Real World*, Prentice Hall PTR 1st edition.
- Computer Dictionary, September 2005, at [http://www.webopedia.com/TERM/W/Web\\_services.html](http://www.webopedia.com/TERM/W/Web_services.html)
- Cooper C.M., Lambert D.M. and Pagh J.D., (1997) *Supply Chain Management: More Than A New Name For Logistic*, The International Journal of Logistic Management, Vol 8,

- Ellram L.M. and Cooper C.M., (1990) *Supply Chain Management and the Shipper – Third Party Relationship*, International Journal of Logistics Management, pp. 1 - 10
- Ethan C., (February 2002) *Web Services Essentials - Distributed Applications with XML-RPC, SOAP, UDDI & WSDL*, O'Reilly & Associates
- Fenton N. E. and Neil M., (1999), *Software Metrics: Success Failures and New Directions*, The Journal of System and Software, vol. 47, pp. 149 – 157
- Gamma E., (1995), *Design Pattern – Elements of Reusable Object Oriented Software*, Addison Wesley.
- Gerstbach P., (2005), *ebXML vs. Web Services - Comparison of ebXML and the Combination of SOAP/WSDL/UDDI/BPEL*, Vienna University of Technology
- Gilani S.F. et al., (2002 ), *Visual Basic .NET Reflection Handbook*, Wrox Press Ltd.
- Hetzel B., (1988), *The Complete Guide to Software Testing*, 2nd edition, John Wiley & Sons Inc.
- Hitz M., Montazeri B., (1995). *Measuring Coupling and Cohesion In Object-Oriented Systems*, ISACC 1995. Monterrey, Mexico (10 p.)
- Houari A. S., Robert G., Thierry M., (September 2005). *Can Metrics Help Bridging the Gap Between the Improvement of OO Design Quality and Its Automation?* at <http://www.iro.umontreal.ca/~sahraouh/papers/ICSM00.pdf>
- Hugo H. and Allen B., (2004), *Web Services Glossary*, at <http://www.w3.org/TR/ws-gloss/>, W3C Working Group
- IBM, J2EE vs .NET, October 2005 at [http://www-06.ibm.com/software/smb/na/J2EE\\_vs\\_NET\\_History\\_and\\_Comparison.pdf](http://www-06.ibm.com/software/smb/na/J2EE_vs_NET_History_and_Comparison.pdf)
- IBM, On Demand Glossary, October 2005 at [http://www-306.ibm.com/e-business/ondemand/us/toolkit/glossary\\_e.shtml](http://www-306.ibm.com/e-business/ondemand/us/toolkit/glossary_e.shtml)
- IBM Web Services Architecture Team, (September 2000), *Web Services architecture - overview - the stage of evolution for e-business*, at <http://www-106.ibm.com/developerworks/web/library/w-ovr/>
- Ines Alves De Queiroz, (2002) *Supply Chain Management: A Business Approach to Some Actual Aspects*. Financial Engineering, E-Commerce and Supply Chain, Kluwe Academic Publishers



- Janyashankar M. Swaminathan, Stephen F. Smith, and Norman M. Sadeh, (1996) *A Multi Agent Framework for Modeling Supply Chain Dynamics*, Technical Report, The Robotics Institute, Carnegie Mellon University
- Kercher, J., (2001) *Authentication in ASP.NET: .NET Security Guidance*, MSDN Magazine August 2001.
- Knudsen D, (1999) *Procurement Performance Measurement System*, Department of Design Sciences, Lund University
- Knudsen D, (2003) *Improving Procurement Performance with E-Business Mechanisms*, Department of Design Sciences, Lund University
- Lee H.L. and Billington C., (1995) *The Evolution of Supply-Chain-Management Models and Practice at Hewlett-Packard*, Interfaces 25
- Li, W. and Henry, S., (1993) *Maintenance Metrics for the Object Oriented Paradigm*, Proceeding of the First International Software Metrics Symposium, Baltimore, MD, May 21-22, 1993, pp. 52-60
- Liberty J. and Hurwitz D., (2003) *Programming ASP.NET, Second Edition*, O'Reilly
- Marciniak J. (ed.), (2001) , *Encyclopedia of Software Engineering, 2nd. Edition*, John Wiley and Sons, pp. 993-1005.
- Martin G., Marc H., Noah M., (June 2003), *SOAP Version 1.2 Part 1: Messaging Framework*, at <http://www.w3.org/TR/soap12-part1/>, w3.org
- Microsoft .NET, October 2005 at <http://www.microsoft.com/net/default.aspx>
- Microsoft SQL Server 2000 Resource Kit, November 2005 at <http://www.microsoft.com/technet/prodtechnol/sql/2000/reskit/part10/c3361.msp?mfr=true>
- Min H. and Galle W.P., (1999) *Electronic commerce usage in business-to-business purchasing*. International Journal of Operations & Production Management.
- Mitchell A., (2002) *Dynamic Coupling and Cohesion Metrics For Java Programs*. Department of Computer Science, NUI Maynooth, Ireland
- Monczka, R., Trent, R., and Handfield, R., (2001). *Purchasing and Supply Chain Management 2<sup>nd</sup> Edition*, South-Western.
- MSDN, Microsoft .NET Framework Developer Center October 2005 at <http://msdn.microsoft.com/netframework/gettingstarted/default.aspx>

- MSDN, Microsoft .NET Framework Developer's Guide, October 2005 at [http://msdn2.microsoft.com/en-us/library/w9fdtx28\(VS.71\).aspx](http://msdn2.microsoft.com/en-us/library/w9fdtx28(VS.71).aspx)
- Neef D., (2001) *e-Procurement: From Strategy to Implementation*, Prentice Hall/Financial Times
- Novack R.A. and Simco S.W., (1991) *The Industrial Procurement Process: A Supply Chain Perspective*. Journal of Business Logistics, Vol: 12, Issue:1
- OASIS, (September 2005) at <http://www.ebxml.org/>
- Orfali R. and Harkey D., (March 1998), *Client/Server Programming With Java and CORBA*, Wiley Computer Publishing
- Roger W., (December 2001 ), *XML Web Services Basics*, at <http://msdn.microsoft.com/webservices/understanding/webservicebasics/default.aspx?pull=/library/en-us/dnwebsrv/html/webservbasics.asp>, Microsoft
- Roman E., Ambler S. W., and Jewell T., (2001), *Mastering Enterprise JavaBeans*, Wiley Computer Publishing.
- Scott W. and Oldfield C., (2004) *The Nine Basic Rules of a Successful Supply Chain*, Waer Systems
- Shyam R. Chidamber, Chris F. Kemerer, (1993). *A Metrics suite for Object Oriented design*, M.I.T. Sloan School of Management
- Sommerville I., (2001) *Software Engineering 6<sup>th</sup> Edition*, Addison Wesley.
- Thomas Connolly, Carolyn Begg, (2002) *Database Systems, A Practical Approach to Design, Implementation, and Management Third Edition*, Addison Wesley
- Tom B., (July 2002), *Understanding UDDI - Tracking the evolving specification*, at <http://www-106.ibm.com/developerworks/library/ws-featuddi/>, IBM
- UDDI.ORG, (October 2004), *UDDI Executive Overview: Enabling Service-Oriented Architecture*, at <http://uddi.org/pubs/uddi-exec-wp.pdf>
- Urban S. D., Dietrich S. W., Saxena A., and Sundermier A., (March 2001), *Interconnection of Distributed Components: An Overview of Current Middleware Solutions*, Journal of Computer and Information Sciences and Engineering, 1(1):23-31
- van Weele A., (2002) *Purchasing and Supply Chain Management, Analysis, Planning and Practice*. Thomson Learning

Vessey, I. and Weber, R., (1984) *Research on Structured Programming: An Empiricist's Evaluation*, IEEE Transactions on Software Engineering, vol. SE-10, pp. 394-407.

W3C Recommendation, Extensible Markup Language (XML) 1.0 (Third Edition), (February 2004) at <http://www.w3.org/TR/>

WordNet, October 2005 at  
<http://wordnet.princeton.edu/perl/webwn?s=methodology>

Yan Y.H. and Klein M., (2005), *Web Services vs. ebXML - An Evaluation of Web Services and ebXML for e-Business Applications*, Faculty of Computer Science, University of New Brunswick, Canada

Zhang Mi, Xu Jianjun, Cheng Zunping, Li Yinsheng, Zhang Binyu, (2005) *A Web Services-based Framework for Supply Chain Management*, Object-Oriented Real-Time Distributed Computing