INVESTIGATION OF INTERNET-BASED SCADA SYSTEMS: DESIGN AND APPLICATIONS

BY

Ramadhan Alaaddin Nouraddin Fan

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SYSTEMS ENGINEERING

June 2004

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Ramadhan Alaaddin Nouraddin Fan** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE IN SYSTEMS ENGINEERING.

Thesis Committee

Dr. Onur Toker (Chairman)

Dr. Lahouari Cheded (Co-Chairman)

Dr. Moustafa El-Shafei (Member)

Dr. Umar Al-Turki
Department Chairman

Dr. Osama A. Jannadi
Dean of Graduate Studies

18/8/2004
Date

To my parents

ACKNOWLEDGMENT

First and foremost, all praise is due to Allah, the Almighty, Who gave me the opportunity, strength, and patience to carry out this work.

Acknowledgment is due to King Fahd University of Petroleum & Minerals, and the Systems Engineering Department, for their support of this research work.

Special thanks go to Dr. Lahouari Cheded and Dr. Onur Toker, thesis advisors, for their patient guidance and generous support for this research, and to Dr. Moustafa El-Shafei for his valuable advice and helpful remarks. I would also like to thank all faculty and staff members who provided support for this research.

I would like to express my appreciation to my colleagues at Saudi Aramco for their support and encouragement, and for sharing the wealth of information that enriched the content of this research.

Finally, I would like to express my sincere gratitude to my mother for her continuous moral support, my father for his valuable comments on this research, my wife for her constant support and patience during those difficult days, and for all friends and family members for their concerns and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

Name:	Ramadhan Alaaddin Nouraddin Fan
Title:	Investigation of Internet-Based SCADA Systems: Design and Applications
Major Field:	Systems Engineering
Date of Degree:	June 2004

The Internet brings many new features to the process control and automation field, which were previously difficult or costly to implement in traditional control systems, such as remote accessibility to the plant, information sharing, and software standardization. The Internet, however, has its limitations when compared to a traditional control system in terms of functionality, performance, security and reliability. Recent emerging web technologies are promising to overcome many of these limitations, and are helping the Internet to evolve into a highly graphical, interactive and collaborative environment.

In this thesis, we investigate the design and applications of Internet-based Supervisory Control & Data Acquisition (SCADA) systems. SCADA systems are typically distributed in nature and are more suitable for this investigation than other types of control systems. First, we provide an overview of what is currently being done to implement Internet-based SCADA systems and compare them to the traditional SCADA systems. Then we discuss the latest web technologies and analyze their uses in industrial automation. Finally, we evaluate how certain web technologies, namely Java and XML, can improve Internet-based SCADA systems. This is done by proposing a software design for a web-integrated SCADA system using the standard UML design approach.

Master of Science Degree
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia
June 2004

ملخص الرسالة

الاســـــــــــم : رمضـان عـلاءالـديـن نـورالـديـن فـــان
عنوان الرسـالة : دراسـة في تصميم وتطبيقـات نظـم التحـكم الرقـابي واكتسـاب البيـانات المعتمـدة على الانتـرنت
التخـــــصـص : هـنـدســة الـنـظـــم
تاريخ التخــرج : ربيع الثاني 1425هـ

توفر شبكة الانترنت العديد من المزايا التي يمكن استغلالها في مجال الأتمتة والتحكم الصناعي، والتي كان من الصعب أو المكلف تطبيقها في السابق باستخدام نظم التحكم التقليدية، مثل إمكانية الدخول إلى المعامل عن بعد، مشاركة أكثر من جهة في استخدام المعلومات، وتطبيق الأساليب المعيارية في البرمجة. لكن استخدام الانترنت مصاحب بجملة معوقات تجعل منه أقل كفاءة وأداءًا وأمناً واعتمادية من نظم التحكم التقليدية. هذه المعوقات أصبح بالإمكان تخطيها مع بدء ظهور تقنيات جديدة على الشبكة العنكبوتية والتي من شأنها المساهمة في تطوير بيئة الانترنت لتصبح أكثر بيانية وتفاعلية وتعاونية.

نقوم في هذه الرسالة بدراسة أساليب استخدام الانترنت في تصميم وتطبيق نظم التحكم الرقابي واكتساب البيانات المعروفة باسم SCADA. هذا النوع من نظم التحكم يعد الأكثر ملائمة لهذه الدراسة نظراً لطبيعته الموزعة. نستعرض أولاً الجهود المبذولة حالياً لتطبيق نظم التحكم المعتمدة على الانترنت ونقوم بمقارنتها بنظم التحكم التقليدية. ثم نناقش آخر التقنيات المستجدة على الشبكة العنكبوتية وندرس إمكانية استخدامها في الأتمتة الصناعية. وأخيراً نقيّم قدرة بعض هذه التقنيات، وبالأخص تقنيات الجافا والـ XML ، في تحسين نظم التحكم المعتمدة على الانترنت. نثبت ذلك بعمل تصميم مقترح لنظام تحكم رقابي واكتساب بيانات مدمج عن طريق الانترنت باستخدام لغة النمذجة الموحدة UML.

درجـة الماجستير في الـعـلـــوم
جـامـعـــة الـمـلك فـهـــد للـبـتـــرول والـمـعـــادن
الظهران، المملكة العربية السعودية
ربيع الثاني 1425هـ

CHAPTER 1

INTRODUCTION

The use of the Internet in process control and automation industry has been driven by two main forces: (1) the commercial trends in computer and information technology fields, and (2) the needs of the process and manufacturing industries. Whenever a match occurs between these two driving forces, companies try to capitalize on the available technology to find technically and economically feasible solutions for industrial problems. This correlation between computer technology and control applications has been historically evident in the design of many traditional process control systems.

The Internet brings many new features to the process control and automation field, which were previously difficult or costly to implement in traditional control systems, such as remote accessibility to the plant, information sharing, and software standardization. The Internet, however, has its limitations when compared to a traditional control system in terms of functionality, performance, security and reliability. Nevertheless, recent emerging web technologies are promising to overcome many of these limitations, and are helping the Internet to evolve into a highly graphical, interactive and collaborative environment.

In this chapter, we provide a historical overview of how the process automation field evolved, how computer control systems are becoming the norm for monitoring and controlling a process plant, and how the Internet will play a major role in this continuing evolution. This is followed by a literature review of previous work in this field, a statement of the objective of this thesis work, and finally a summary of the organization of the remaining chapters in this report.

1.1 History of Process Automation

In the 1960's, most industrial process automation was done pneumatically. Air pressure, typically in the range of 3-15 psi, was used for communication and control. The process variable was measured and converted to an air pressure signal by a pneumatic transmitter, the transmitter sends the air pressure signal through a tube to a receiver, and the receiver serves as an indicator, controller, recorder, or relay. Pneumatic actuators were used to control the final control elements. Pneumatic relays were used for logic functions, such as high or low select, as well as simple arithmetic functions, such as addition, multiplication, square root, and totalization. All working parts within pneumatic instruments were mechanical [1, 2].

Pneumatic control systems have been used for many years due to the fact that their components are inexpensive, reliable, and intrinsically safe. This was critical for applications that existed in explosive environments. Pneumatic instrumentation was also more resistant to corrosive environments. However, one of the main disadvantages of pneumatic signal transmission is the problem of transmitting the signal over significant distances. A time lag is associated

with the transmission of the pressure signal through the instrument tubing. As the distance that the signal must travel increases, the speed of the response of the pneumatic transmission systems increasingly becomes a problem for fast loops such as flow.

With the introduction of the transistor, many pneumatic devices were replaced by electronic counterparts. Electronic signals convert the process or control variable to an analog electric signal, usually a voltage (1 to 5 volts) or a current (4 to 20 mA). Electronic signal transmission has a virtually instantaneous time response and hence nearly unlimited distances were achievable by wire, radio linkage, or microwave signals. However, electronic control systems had the disadvantage of becoming very complex for large applications. Therefore, it was not recommended to use electronic instruments except for simple control loops or logic functions.

The advent of the microprocessor in the late 1960's has revolutionized the process control industry. Digital signals were used to approximate analog signals over time. The electronic analog signal is sampled at a predetermined frequency with the value of the last sample held until the next sample is taken. Analog measurements are converted to digital signals through an analog/digital (A/D) converter, and control outputs are converted back from digital to analog (D/A) before being used by the field devices. Digital controllers have tremendous control flexibility compared to its pneumatic or electronic counterparts. They can be used for stand-alone single-loop control, or they can be linked together on a common data highway to provide distributed control capabilities.

Although microprocessor-based control systems started in the late

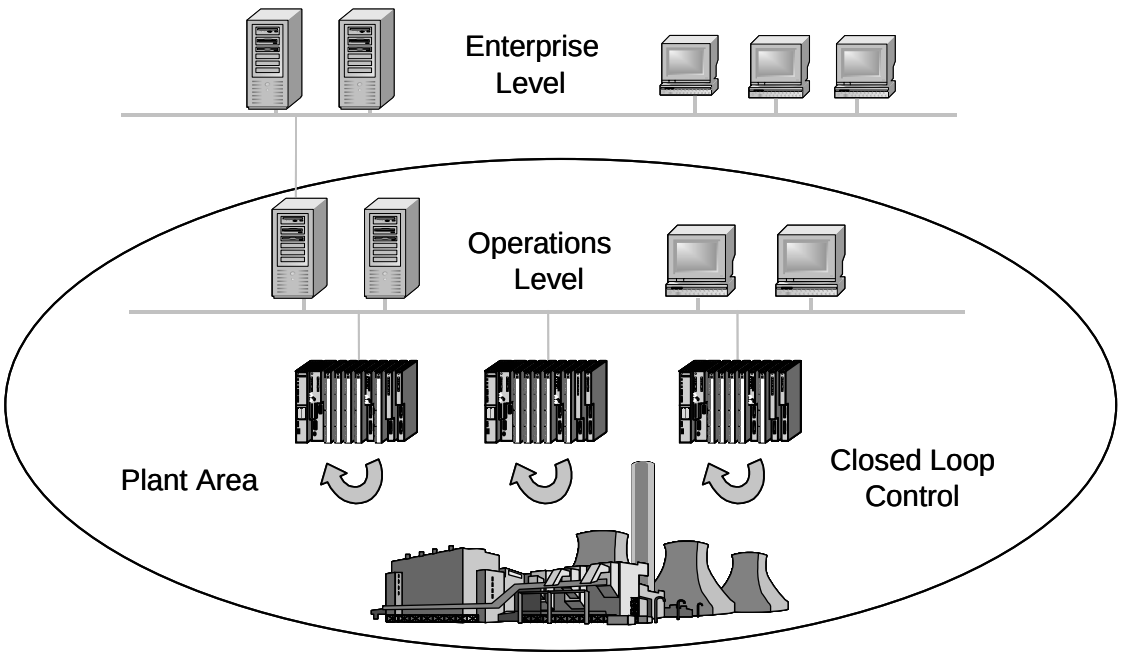


Figure 1.1: Distributed Control System (DCS)

1960's, they were used merely to supervise the classical pneumatic or electronic systems. It was not until the mid 1970's that the first Distributed Control Systems (DCS) came into existence. A DCS system (Figure 1.1) distributes the functions of a control system into many different microprocessors. The microprocessors form small subsystems which are physically distributed throughout the facility and linked together via a communication or data highway. The operator interface is located in a central location such as a control room. The operator interface includes color graphics with dynamic process data, faceplate displays, trend data, and an alarm summary or annunciator displays.

For discrete processes, hardwired relays were used to perform the sequential logic control functions. Electromechanical relays were wired in series or parallel with field input signals to turn equipment on and off. Input devices such as pushbuttons, switches, or contacts were used to allow current to flow through the circuit or cause a break in the current flow, thereby switching an instrument on or off. Instruments that were controlled by relay logic outputs were coils, timers, lights, valves, horns, and control relays. The main disadvantage of these hardwired relays is that expansion was not possible without disabling the system, the control logic could not be changed without adding or deleting relays or disabling the system, and it was difficult to implement and troubleshoot a complex logic system.

Due to the difficulties of hardwired relay systems, a new type of microprocessor-based control system, known as the Programmable Logic Controller (PLC), appeared in the early 1970's in the automotive industry. The PLC systems (Figure 1.2) replaced the hardwired relay panels and provided the

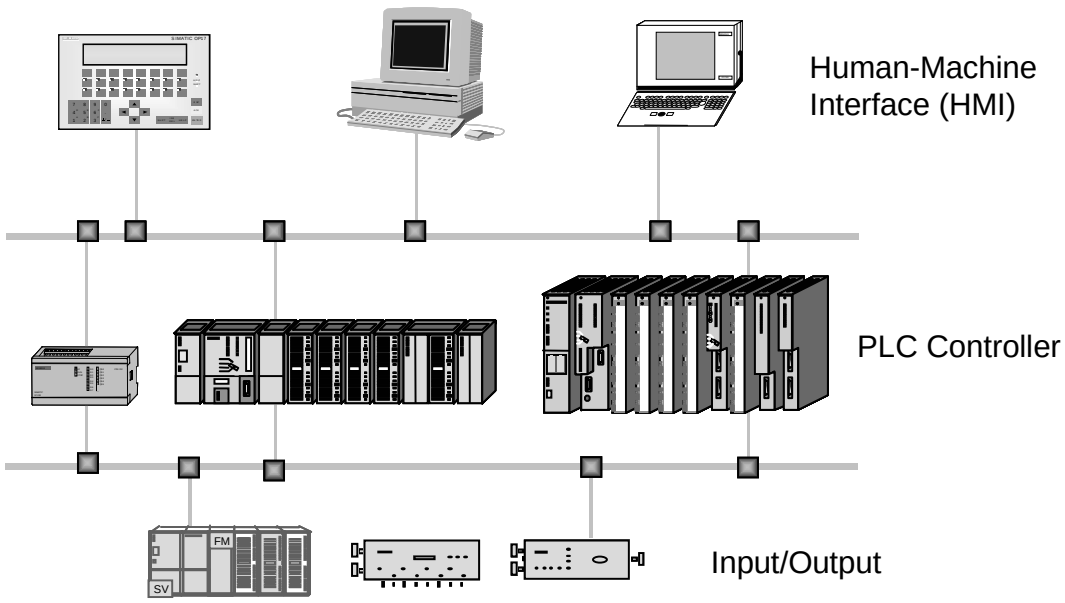


Figure 1.2: Programmable Logic Controller (PLC) System

flexibility of changing the logic program whenever the application at hand changed. PLCs also demonstrated a high degree of reliability at a fairly low cost. That is why they became more and more popular in the process control industry, and they have grown in the market to perform virtually any control application ranging from simple on-off control to relatively complex control applications.

For other types of applications which require communication with various remote sites, the Supervisory Control and Data Acquisition (SCADA) system was introduced, also in the late 1960's. The SCADA system (Figure 1.3) gathers information from remote terminal units (RTUs) that are usually scattered across remote geographical locations, such as across a pipeline or multiple power stations. The SCADA talks to the RTUs through dedicated communications circuits, and monitors the transmission of information in real time. If a leak on a pipeline occurs for example, the RTUs transfer the information back to the central SCADA system, and the necessary analysis and control actions are taken within seconds.

Starting in the 1980's, advances in the field of information technology (IT) and the development of personal computers (PCs) and communication networks such as local area networks (LANs) and wide area networks (WANs) have opened up yet more possibilities for process automation. The use of standard computers and communication networks in the industrial environment enabled the use of ordinary, off-the-shelf products to do the job of custom-built DCS, PLC or SCADA system. PC-based control systems offer users the flexibility and scalability that proprietary vendors could not meet in the past. The distinction

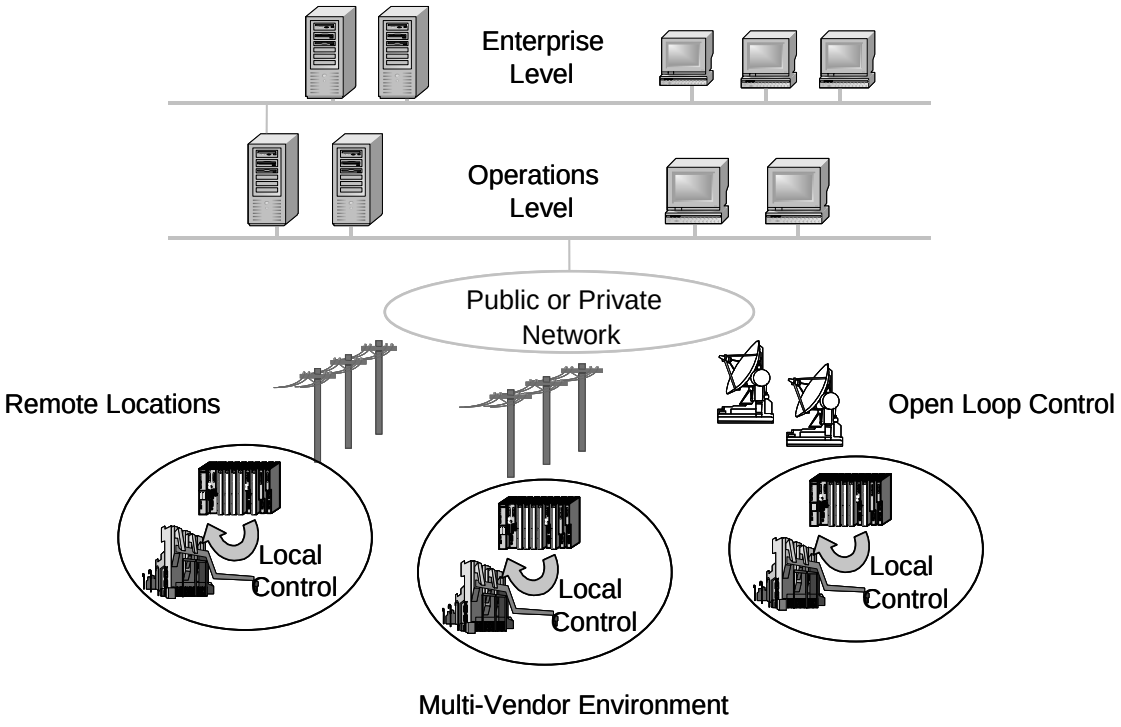


Figure 1.3: Supervisory Control and Data Acquisition (SCADA) System

between these systems is becoming more and more blurry as they begin to adopt each others' functionality and add similar features to satisfy the end user demands and earn additional market share.

1.2 Typical Applications of Computer Control

Today, almost all new instrumentation systems are based upon microprocessor technology, ranging from the small single loop digital controllers to the sophisticated and powerful multi-processor distributed control systems. Computers are typically interfaced to the standard instrumentations to provide more mathematical power for advanced control applications and optimization routines which are not available in the base level instrumentation. Computers also provide more sophisticated operator interface and display capabilities which can be used to monitor process variables, process calculated variables, send setpoints or control commands, and generate reports and historical data trends.

Because computer control systems are less reliable than the redundant fail-safe instrumentation systems, they are not directly connected to the process. They are interfaced with instrumentation systems to provide a fall-back operating position. If the main computer processing fails, the instrumentation systems would resume their work and the plant can still operate without the computer. Typically, the computer control system can either be a centralized processing system, in which only one processor performs all the tasks, or it can be a distributed networked system, with each individual node on the network responsible for a specific application, thus providing more robust control features [1, 2].

1.2.1 Process Monitoring

The purpose of computer-based process monitoring is to gather and view plant and facility information. Process monitoring is used for data acquisition and presentation, not for process control. The monitoring system can calculate items such as reactor yields and economic variables and can display process variables such as reactor temperatures, tank levels, and process flow rates. This information can be displayed to the operator and presented in an easy-to-use form. For example, color graphics can be used to present process alarms and data to operators in a form that can be easily and quickly understood. However, monitoring systems are not limited to use by the operator. The same information can be analyzed, reformatted and presented to engineers and management. This information reflects the present condition of the facility and aids in optimum planning and scheduling.

The traditional process monitoring application is designed for use by the operator. Process alarms and the status of the plant are displayed. Graphics, tables, and plots of plant variables are used to help the operator understand large amounts of data. Various mathematical operations can also be performed to aid the operator. For example, a noisy process signal can be passed through a digital filter. Another common application is the use of a program to estimate a process variable that cannot be measured; an example of this is the on-line estimation of the octane of a gasoline stream from a reforming unit.

Process monitoring systems can be thought of as information managers. Modern process monitoring systems are no longer limited to operator interfaces; they have the ability to distribute real-time information to engineers and

managers. These systems are able to interface with existing DCSs, PLCs, and other sources to accomplish this goal. Management requires up-to-date information for accurate planning and scheduling decisions. Engineering needs both current information on facility activities and historical data for engineering studies. Ideally, a monitoring system will allow all functions access to the facility database and will present the data in a useful format. Management should also be able to pass information back to operations in order to implement their goals.

As technology advances, the differences between a DCS and a process monitoring system become less distinct. However, distributed control systems are primarily designed as process control systems. These systems, including their communications network, are usually fully redundant. The graphics in these systems are very powerful, but they are directed toward operators and process control. The same is true of data management with a DCS; engineering studies can be performed and plant reports can be generated, but these tasks can be more easily accomplished by process monitoring systems. Finally, DCSs are generally expensive. In contrast, process monitoring systems are directed more toward information management. These systems are not as critical to plant operations and hence are not fully redundant. They are relatively inexpensive.

1.2.2 Continuous Process Control

A continuous process is one in which process material is continually flowing through the process equipment. Continuous process control involves the continuous measurement of a process variable via an analog signal and the adjustment of a final control element, such as a control valve to keep the

process measurement at a desired value. Process values are maintained close to their targets or setpoints despite changes in the process or process upsets. Disturbances in the process caused by changes in, for example, feed composition and rate, fuel gas composition, or pressure are kept to a minimum.

Computer control systems for continuous processes use either direct or supervisory control. With direct control, the control algorithm calculation (such as PID) is done within the computer, and the computer then sends the valve position signal directly to the control valve through the controller. In supervisory control, the setpoint is calculated in the computer and then sent to the controller. The controller then performs the control algorithm calculation and determines the proper valve position. Many of the original computer control systems used direct control because the early instrumentation was not designed for supervisory control. However, due to the slow response of computer control systems compared to instrumentation systems, the need for supervisory control arose.

Usually, server-level computers provide the mathematical power for some of the advanced control functions and optimization routines. PCs or workstations are used for monitoring and operator interface, or for relatively simple control applications. A common practice is to use PCs interfaced with single loop controllers to perform supervisory control functions. PCs are generally not reliable enough for standalone direct control or for large complex processes.

The DCS controllers are typically responsible for a large number of control loops and have a large input/output (I/O) count. Because an individual

controller is responsible for a large portion of the facility, backup systems have been developed to prevent catastrophic events upon DCS hardware failure. Many of the functions within the system are redundant and have automatic backup upon failure. All control functions and most input and output (I/O) processing is capable of being made either redundant or fault tolerant. Almost all continuous control applications of significant size are using distributed control systems. Those applications requiring special or dedicated systems can be interfaced to the DCS, allowing the operator a single window to the process.

1.2.3 Discrete Process Control

Discrete or sequential control is often referred to as on/off control. It is a series of discrete control actions performed in a specific order or sequence. These actions can be the opening or closing of valves or the starting or stopping of devices. The control actions can be initiated by an operator or a process condition or as a result of the passage of a given period of time. Sequential control applications include pipelines, water treatment, utilities, effluent processing, and packaging.

Discrete control computers are used above and beyond the standard data acquisition and control system such as the PLC. The type of computer system used for discrete control systems are typically the Supervisory Control and Data Acquisition (SCADA) systems. SCADA systems are computer systems that allow the control and monitoring of a process in remote areas from a centralized location. From this centralized location an operator can view the entire process through graphic and trend displays and initiate commands and setpoint

changes to the field. SCADA systems are mainly dedicated to monitoring and discrete control applications, but they can also perform some simple continuous control functions.

The distance between the centralized location and the individual process can be quite large, covering many miles. Communication is therefore a key ingredient of a SCADA system. Because SCADA systems are capable of passing data over large distances, they fulfill the application needs of the pipeline and producing platform operations. Particular applications include leak detection, batch tracking, well tests, tank monitoring, inventory reports, and meter proving.

1.3 Use of Internet in Process Automation

One of the most obvious advantages of the Internet is the remote accessibility of plant systems and the sharing of this information among various people in the organization. Another advantage is the distributed open-system architecture that the Internet provides and hence allowing heterogeneous systems to communicate with each other. A third advantage is the use of the standard web browser, which provides a uniform human-machine interface (HMI), hence minimizing maintenance and training costs. All these advantages led many control system vendors to offer web-enabled versions of their traditional control systems, or in some cases, provide complete web-based solutions for certain applications.

On the other hand, limitations of the Internet arise from the fact that Internet applications have been historically based on textual web pages and

transactional databases; whereas control systems require interactive graphical displays with real-time dynamic data. The functionality and performance constraints, in addition to concerns inherent to the Internet such as security and reliability, limit the widespread use of web-based control systems.

Most of the implementations so far utilize the Internet (or intranet) as a medium for communication and plant management information only. Few have tried to control a process remotely, or implement what can be called an Internet-based control system. A typical web enabled plant monitoring application (Figure 1.4) may be limited to converting DCS displays into web pages then publish them through a web server. The user accesses the web server to view the displays remotely using a normal web browser from any general-purpose PC. The user may have additional features such as trending, reporting and interfacing with other desktop tools.

The Internet can be interpreted in two ways: it can be viewed as a communication medium that is composed of inter-connected networks (actually this is where the name came from); or it can mean the Web, where you have web servers providing the services and the browsers as the clients. In both cases, TCP/IP is used as the networking protocol, and the physical connection and data link are established through standard topologies like Ethernet, Token Ring and ATM. Many of today's process automation systems are designed to operate on one of these standard network technologies. Therefore, many of these systems can be easily connected to the Internet/Intranet provided that physical connection can be made between the plant's LAN and the corporate WAN.

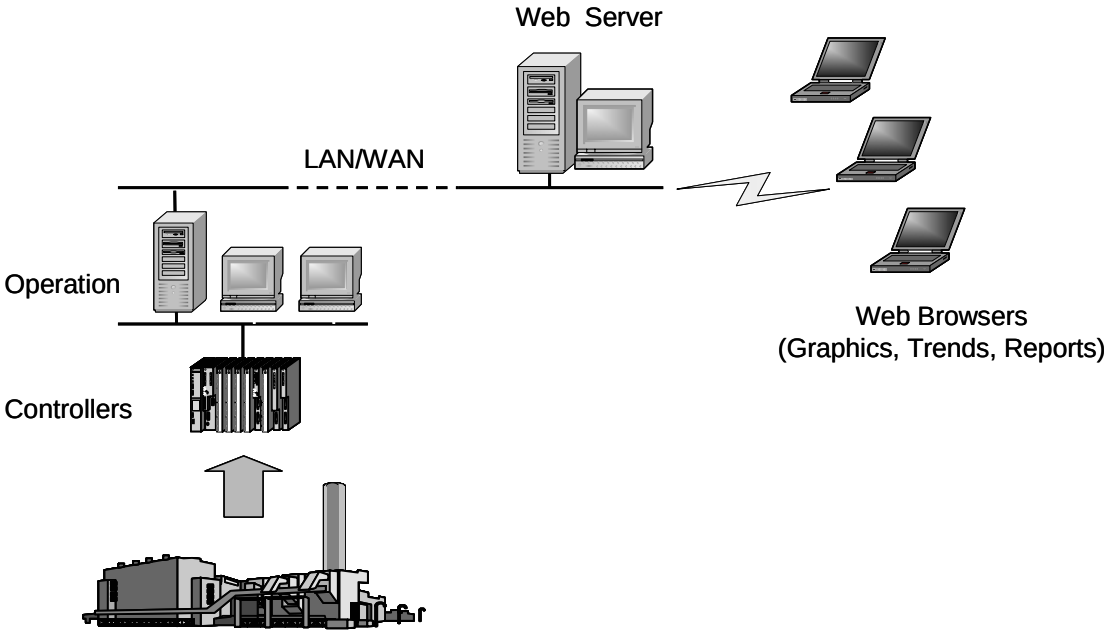


Figure 1.4: Web enabling of a process control system

This connectivity, however, only satisfies the lower four layers of the OSI networking model. In order to establish communications on the upper layers, applications must know how to talk to each other over the network. This has been the major obstacle in process system integration so far, and the normal practice to overcome this problem has been to write custom drivers and develop code to link between the various systems. The same practice was also used to link between the various devices within a plant system. This is all beginning to change with the use of object-oriented software.

The de facto communication standards are playing a major role in process automation. Ethernet and TCP/IP are becoming the network standard because of their cost-effectiveness, cross-industry pervasiveness, and natural compatibility with web-based user interfaces. Also, object-oriented software are promising easy integration between various systems without the need for custom drivers or code. It is now possible to define devices, people, and forms and how they should act with each other in the context of larger manufacturing and business processes.

New developments in e-commerce are providing more alternatives for process automation through the web. The web services concept offer new tools for enterprise integration. Microsoft's .NET framework and Sun Microsystems' Java2 Enterprise Edition (J2EE) are both based on XML (Extensible Markup Language) which allows applications to communicate and share data over the Internet, regardless of the operating system or programming language.

In addition, advances in database technology also play a major role in real-time data communication. Vast quantities of real-time information can now

be defined and captured efficiently. They can also be easily distributed (mirrored) to all the various points on the network where it is needed for analysis, processing or interaction. Web-based real-time data streaming from field units can be visualized in real-time in customized user interfaces. For applications that do not require constant real-time data streaming, data can be viewed on demand.

1.4 Literature Review

The concept of monitoring and controlling a process via the Internet and using a standard web browser is not a new idea. It has been experimented with in university laboratories, as well as commercially offered by process automation vendors.

1.4.1 Academic Experiments

Most of the laboratory experiments were driven by the need for establishing "virtual" control labs that can be used for distance e-learning. One of these virtual labs was developed by Dr. Toker and Dr. Al-Sunni of KFUPM System Engineering Department [3]. This project is explained in more detail in Chapter 5 of this thesis report. Other examples found in the literature are mentioned here.

One of these examples is the work done by Sanchez et al. at the Universidad Nacional de Educacion a Distancia in Spain [4]. The system consists of a simulation engine developed in MATLAB, and a control application developed in Java. The Java application is used as a stand-alone control system,

while an equivalent Java applet (which is a small program embedded within a web page) is used for web-based remote control. In both cases, the system maintains a clear separation between the math and simulation engine (MATLAB) and the graphical experimentation interface (Java). This way, the math engine can be replaced with a different one or with a real plant.

Another virtual lab was developed by Ramakrishnan et al. at the National University of Singapore [5]. In this system, a physical experimentation apparatus (dual tank system) is used for process modeling. The lab instruments are interfaced to personal computers that are connected to the Internet. The control application is developed using National Instrument's LabView software, Java programming language, HTML (Hyper Text Markup Language), and other scripting languages like JavaScript and VB script. The implementation also uses video conferencing to provide fast and point-to-point visual feedback to the remote user.

The use of Java and HTML in developing web applications is common since they are the main programming languages for the Internet. Java was introduced by Sun Microsystems in 1995 and is mainly used for developing application programs that are distributed and portable. Java programs can be written and compiled on one computer and loaded and run on another regardless of the hardware or operating system of these computers. This "platform-independence" has led to Java being often called “portable code”. One common way to implement Java in web applications is through Java applets embedded in web pages.

HTML was introduced in the 80’s as a means for linking documents over

a wide area network. Since then, HTML has evolved to become the main language for developing web pages. However, HTML is limited to processing simple (mainly text-based) applications. It was never intended to describe complex documents or carry the weight of the Internet as it does today. The HTML's shortcomings are compensated for by embedding Java applets or other scripted code into the HTML documents. In addition, a new language called XML (Extensible Markup Language) has been introduced by the World Wide Web Consortium (W3C) [6] to describe the content of the web page rather than just the format (as is typically done by HTML). XML is derived from a philosophy that data should be delivered without binding to a particular script language. Therefore, XML provides “portable data.”

1.4.2 Industrial Applications

The Java programming language is also being implemented in many industrial process automation systems. This includes both low-level plant floor control as well as high-level enterprise integration [7]. An example of plant floor control has been demonstrated by Khavar of Cyberonics [8], where a simple Java-based control system is used for machine positioning, batch processing, manufacturing execution and remote I/O operations. The system uses a PLC running Java software objects, and the HMI software developed using Java language.

Morgenthal [9] describes how XML works with Java to provide "portable data and portable code". This combined portability has many implications for the process automation industry. For example, process data can be exchanged

and shared between heterogeneous systems within a company; process data can be manipulated and presented in specific personalized formats to different users; process applications can be integrated and fully automated for enhanced operational efficiency; process applications can be deployed on any platform and integrated easily within the existing infrastructure, hence involving minimum increase in development and implementation cost.

The W3C organization also specifies technologies for Web Services, which is a new phenomenon aimed at restructuring the World Wide Web into a collaborative environment to better serve E-commerce. The web services environment allow applications to integrate with each other using XML. Two major web service platforms are available: Sun Microsystems' J2EE (Java 2 Enterprise Edition) which is based on the Java Beans technology, and Microsoft's .NET platform which is based on the ActiveX/COM technology. Vawter and Roman [10] provide a comprehensive comparison between J2EE and .NET.

Most of these E-commerce technologies are finding their way into the manufacturing and process industries. In an article published by ISA (Instrumentation, Systems and Automation Society), Pinceti [11] states that technology usually moves from information systems into industrial automation within three to five years. Pinceti discusses how a new paradigm is being introduced with the use of PC-based control systems. This new paradigm splits into several levels, starting from the base automation devices (instrumentation and actuators in the field), to local control units (PLC's, Intelligent I/O's, PC-based units), to area control systems (networked PC's, operator HMI's), and

finally to the communication and information systems (corporate operational, financial, and decision-making systems).

The techniques used for communication between these automation levels include the IEC 61158 Fieldbus standard (used for the lower levels) and the IEEE 802.3 Ethernet standard and TCP/IP protocols (used for the higher levels). The information exchange is done via various communication protocols such as OPC (OLE for Process Control) which employs Microsoft's COM (Component Object Model), and CORBA (Common Object Request Broker Architecture) which provides an alternative for non-Microsoft environments. Both protocols are specific to process control and are thus limited when it comes to general-purpose system integration. This is why there is an increasing trend to use standard Internet communication protocols (such as HTML and XML).

In another ISA article, Bono [12] discusses the reasons behind adopting E-commerce technologies in manufacturing industries. The main reason for this is the embracing of XML as a simple method to integrate between different systems, from different vendors, which is typical of process systems existing within a company. Additional reasons include lower prices, more flexibility, easier data access, new applications and automated maintenance. Bono also lists the challenges to XML ubiquity, such as the need for standardization, embedded systems support, discovery mechanisms and maintenance tools.

The benefits of using the Internet in industrial automation have been highlighted by various automation vendors. This is usually promoted in the sense of process remote monitoring, information sharing, and enterprise integration, more than being a means for process control and operation.

Nevertheless, some vendors do offer complete Internet-based SCADA solutions, like vMonitor. Their products consist of web-enabled field communication devices and Java-based software. The remote control is based on control algorithms provided or specified by the client.

Gunst and Stein of Intellution [13] list the benefits of using Internet in manufacturing and predict a wide spread of this phenomenon based on a Forrester Research report issued in 1997 which forecasted that Internet computing will be used for solving industrial problems, all desktops will have browsers, web content will be dynamic and system integration methods will be based on standard protocols and components rather than custom code. Kennedy and Eisele [14, 15] describe how the industrial desktop PC's evolved over time to become complete web-based solutions. The term "industrial desktop" has been coined by Kennedy of OSIsoft [16] to describe how computers can be used to manage complex manufacturing facilities in the process industries.

A new trend is also being followed by some automation vendors (mainly in the US) and that is to provide Internet-based "services" for process automation using an Application Service Provider (ASP) model. This model includes a web-hosting service centrally located and managed by the ASP, and remotely-located customer systems which are integrated with the ASP central system. Heersink and Wright of Industrial Evolution [17, 18] describe their ASP solution for an oil refining company, where live process data are collected from the operating plants and hosted on the Internet or intranet for remote web access. Diehl and Moyes of Matrikon [19] describe their SCADANet solution for a midstream oil

and gas company.

The advertised benefits of using the Internet in industrial automation do not come without their associated risks. As Bailey of Ci Technologies [20] warns, there are several considerations that need to be taken when using an Internet-based SCADA system. This includes, for example, security concerns which are typically overcome through firewalls and password protection. For real-time applications, a single level of security is usually not sufficient. Several systems implement what can be called point-by-point security, where each user is given access to specific data only.

Also, functionality restrictions come from the fact that web browsers are limited to HTML or XML functionality, providing limited support for pop up windows, and dynamic allocation of data links or memory. To get a suitable functionality that is close to a SCADA's, the browsers need a significant number of plug-ins to be loaded on the client, which has impacts on performance and maintenance. Performance is also critical in process control systems especially with the large number of graphics used. Even with large-bandwidth systems, the client must be responsible for most of the graphics handling and database update.

The performance of the network environment in which the SCADA system is running plays a major role in the performance of the overall system. In an analysis done by a company called DCB (Data Comm for Business) [21], they conclude that the Internet, as it operates today, is not suitable for SCADA applications since it has variable delays and can suffer packet loss that can be as high as 10% or more during severe congestion. They believe that the Internet

may be acceptable for the retrieval of data on an occasional basis but not for real time data collection and control requirements. They recommend using high-speed networks and more reliable communication protocols (e.g. frame relay, Ethernet, IP networks) to ensure better performance of the SCADA systems.

1.5 Objective of the Thesis

The objective of this thesis work is to investigate the design and applications of Internet-based process control systems. Figure 1.5 shows the overall research plan. Part 1 analyzes the advantages and disadvantages of Internet-based SCADA systems in comparison to traditional SCADA systems. SCADA systems are typically distributed, remote, and multi-vendor in nature and are thus more suitable for this investigation than other types of control systems. Part 2 discusses the latest web technologies that have emerged on the Internet and analyzes their uses in the e-business world, as well as its implications for the process automation world. This is then followed by our research hypothesis in part 3, which is to evaluate how certain internet technologies, namely Java and XML, can improve the design of current Internet-based SCADA systems.

The hypothesis is tested by looking at two projects. The first was conducted separately from this thesis work and it uses a lab-scale Java-based control system to control a typical process control application (such as a dual tank system). The second is a design project conducted as part of this thesis work and it extends the preceding work to show, in standard unified modeling language (UML) notation, how separate control systems can be integrated with each other, using Java and XML, to achieve a wider distributed control function

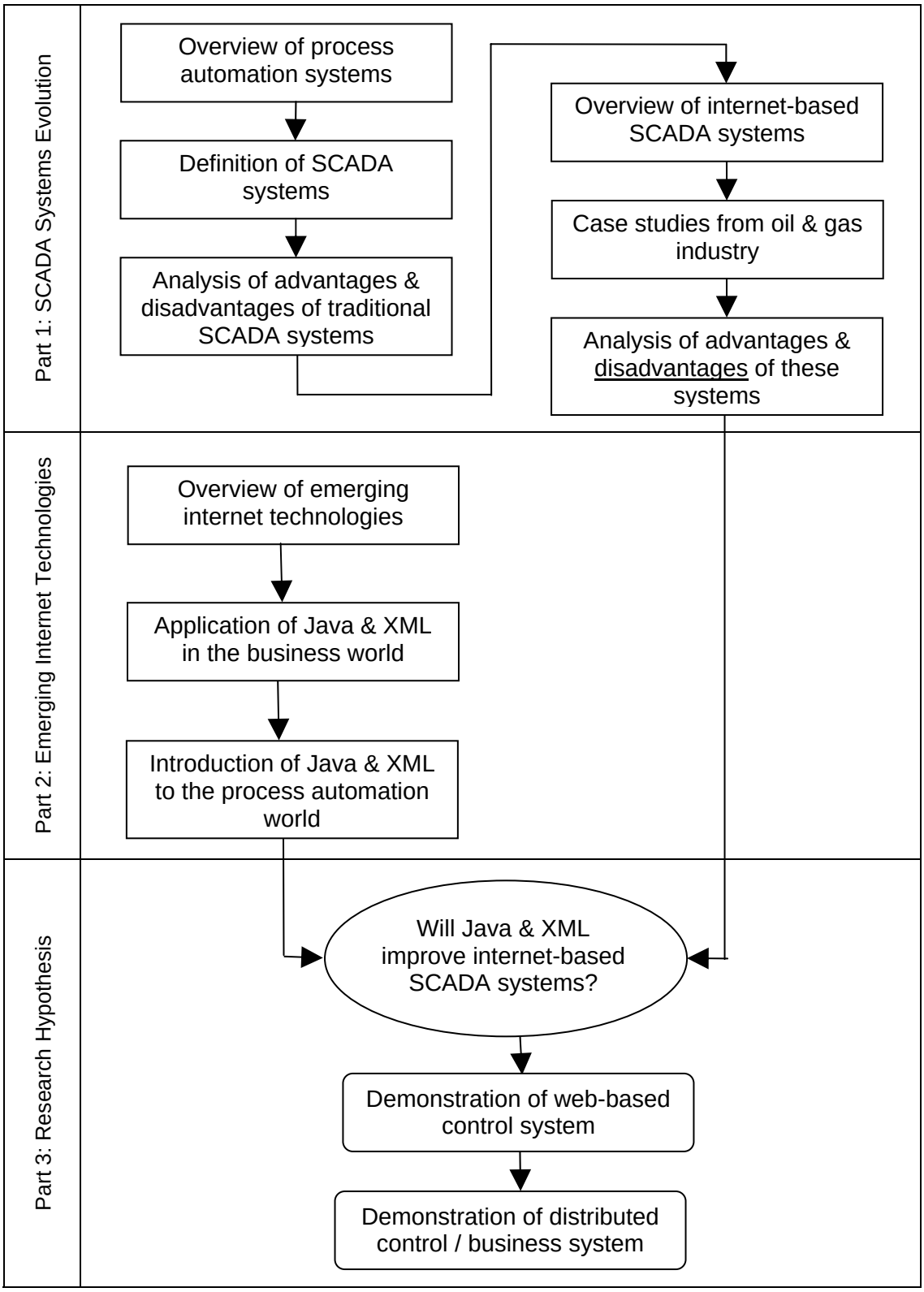


Figure 1.5: Research plan

By combining the user's and developer's perspectives, the thesis will generate an unbiased report that will explain what is currently being done in the area of Internet-based process control, how it is done, and how it can be improved upon in the future. This will provide some guidance for local process industries (such as Saudi Aramco and SABIC which are widely distributed within the Kingdom) to understand the design of web-based process control systems, identify their advantages and disadvantages, and eventually select the control solution that is most suitable for their needs.

1.6 Organization of the Report

The remainder of this report is organized as follows. Chapters 2 and 3 trace back the evolution of SCADA systems, from the traditional ones to the Internet-based, and analyze the advantages and disadvantages of each. Chapter 4 looks at emerging Internet technologies and how they are applied to the business world. Specifically, it focuses on two web technologies, Java and XML, and discusses how combining both of them together can be advantageous. Chapters 5 and 6 test the research hypothesis by designing web-enabled and web-integrated control systems, respectively. Finally, chapter 7 summarizes the findings and results.

CHAPTER 2

TRADITIONAL SCADA SYSTEMS

In this chapter, we provide an overview of traditional SCADA systems, how they are used, what type of processes could benefit from them, and what key elements constitute a SCADA system. Then we discuss how these traditional SCADA systems are evolving over time and starting to adopt the industry standards for communications and integration. Finally, we provide some insights into the implementation side of SCADA systems, what costs are usually associated with a typical SCADA system project, and what are the disadvantages and implementation concerns. These discussions will become apparent as we move to the next chapter which explains the design and applications of Internet-based SCADA systems.

2.1 Definition of SCADA

Supervisory Control and Data Acquisition (SCADA) is a term used to describe the technology that enables a user to collect data from one or more distant facilities and send limited control instructions to those facilities [22]. A SCADA system is an industrial measurement and control system consisting of a central host or master (usually called a master terminal unit or MTU), one or more field data gathering and control units (usually called remote terminal units or RTUs),

and a collection of standard and/or custom software used to monitor and control the remote field data elements.

SCADA systems make it unnecessary for an operator to be assigned to stay at the remote facilities or frequently visit them when everything is operating normally. The SCADA system will alert the operator if alarm conditions are present and will allow him to take the proper actions. Traditional SCADA systems exhibit predominantly open loop control characteristics and usually utilize long distance communications, although some elements of closed loop control and/or short distance communications may also be present.

Traditional SCADA systems are similar to DCS (Distributed Control Systems) in that they are both used for process control and monitoring. However, they differ in that with DCS systems, the field data gathering or control units are usually located within a more confined area (e.g. factory, refinery, power plant) and the communications are usually done via a reliable and high speed local area network (LAN). The SCADA systems on the other hand generally cover larger geographic areas (e.g. pipeline, power lines) and rely on a variety of communications systems that are normally less reliable than a LAN. DCS systems also employ significant amounts of closed loop control, while SCADA systems use mostly open loop control due to the less reliable communication.

A traditional SCADA system performs data acquisition functions by scanning field inputs at the RTUs, communicating those field inputs to an MTU through public or dedicated communication links, and then processing those inputs at the MTU. The traditional SCADA system will also perform automatic

control, or manual control initiated by the operator, by sending the command signals to the RTUs via the same communication links.

The human-machine interface (HMI) for the operator is typically done through graphical displays which show a representation of the plant or equipment under control. Live data can be shown as dynamic graphical shapes over a static background. As the data changes in the field, the foreground is updated, either as digital states (valve open or closed) or analog values such as numbers, bars or charts. Control elements are shown as buttons or set-point values.

2.2 Applications of SCADA Systems

2.2.1 Applications Suitable for SCADA

SCADA technology is best applied to processes that are spread over large geographical areas, are relatively simple to control and monitor, and require frequent, regular, or immediate intervention. SCADA systems have been successfully installed on many processes with a wide variety of applications, ranging from simple on/off control to more complex and advanced control. Although SCADA systems limit the amount of control that can be exercised remotely, they still do allow control. This is one of the things that distinguish SCADA from most telemetry systems. The complexity of remote control that is possible with SCADA systems has grown as the technology has matured.

Examples of SCADA applications include oil and gas production facilities, such as wells, gathering systems, fluid measurement equipment, and pumps, which are usually spread over large areas. They require relatively simple

controls such as turning motors on and off, need to gather meter information regularly, and must respond quickly to conditions in the rest of the field. Also, pipelines for oil, gas, chemicals, water, and wastewater have elements located at varying distances from a central control point. They can be controlled by opening and closing valves or starting and stopping pumps, and must be capable of responding quickly to market conditions and to leaks of dangerous or environmentally sensitive materials.

Other examples include electric transmission systems which may cover large geographical areas within a city, region, or country. They can be controlled by opening and closing switches, and must respond almost immediately to load changes on the lines. Also, groups of small hydroelectric generating stations that are turned on and off in response to customer demand are usually located in remote areas. They can be controlled by opening and closing valves to the turbine. They must be monitored continuously, and they need to respond relatively quickly to demands on the electric power grid.

2.2.2 Applications Not Suitable for SCADA

There are applications which are simply not suitable for SCADA. This may not have been clear during the early stages of SCADA introduction, but as the technology matured, the hard school of experience proved that Murphy's Law exists. Remote control systems can be counted on to perform flawlessly until that moment when a critical command must be sent or an important piece of data is working its way from one end of the system to the other end. In general, two types of signals should not be designed to depend on SCADA: safety

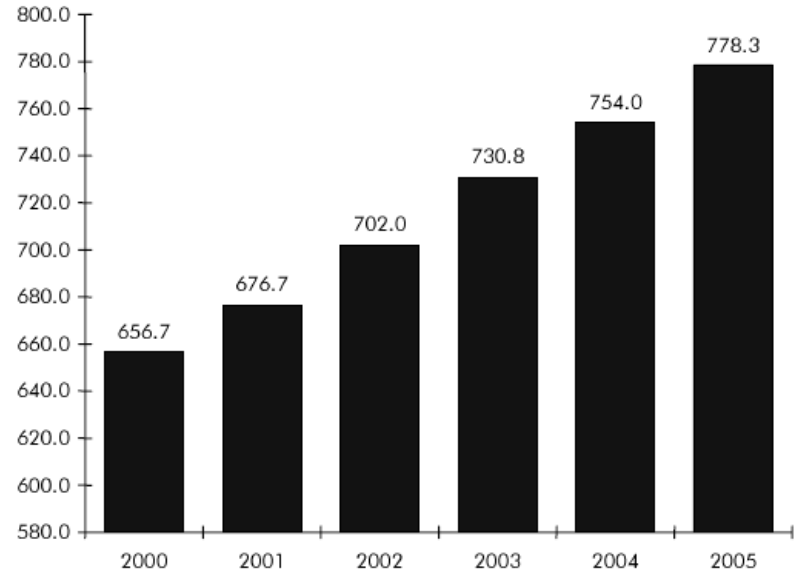
systems and product measurement systems.

Safety systems are designed such that failure in some part of the system will not cause injury to a person or cause damage to the equipment or the environment. They should be designed with minimum number of parts and electrical contacts, and should not share components with normal controls. The sensing, logic, and actuation features of the safety system at a local site should all be self contained. Therefore it should not rely on the SCADA system to send its signals. The SCADA system can be used, however, to enhance the safety of geographically diverse applications such as pipeline leak detection. The pipeline inflow and outflow could be measured, and if the difference between the two is too large, the SCADA system could close the block valves along the line.

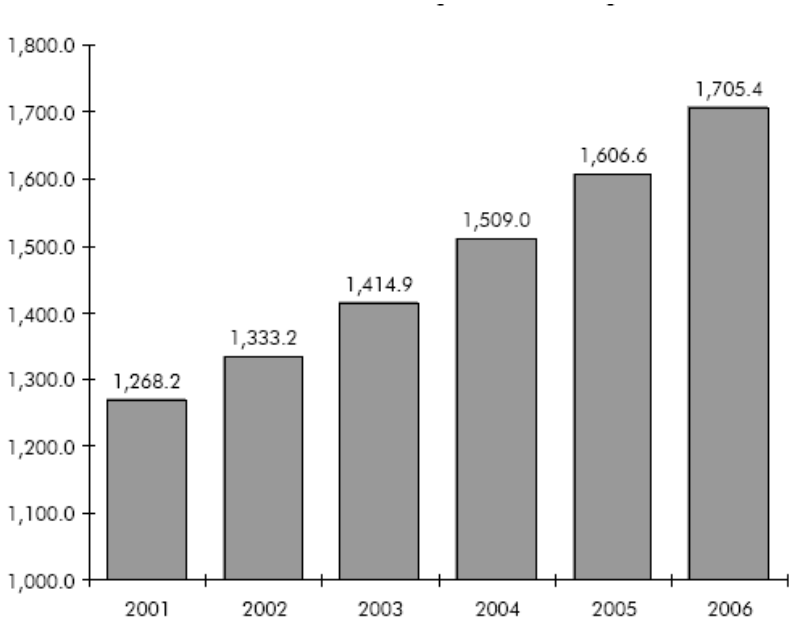
Product measurements systems may involve billing and reporting of royalties, tariffs, and taxes. They may be governed by regulatory bodies which require that measurements be reported to one or more government agencies. An example of this is the measurement of oil from a well which require that a royalty be paid. The accounting and auditing procedure may require paper records to be maintained. This is not going to be supplanted by another method just because SCADA systems are capable of electronically moving some of those data elements.

2.2.3 Industry Demand for SCADA

The ARC Advisory Group [23, 24] has conducted several market analysis studies and technology forecasts for the application of SCADA systems in the industry. Two of these studies are concerned with the outlook of SCADA



(A)



(B)

Figure 2.1: SCADA market in millions of dollars for (A) oil & gas and water & wastewater industries (B) electric power industry (Source: ARC)

systems market, one for the oil & gas and water & wastewater industries, and the other for the electric power industry.

Oil & Gas and Water & Wastewater Industries. According to the first study [23], the market for SCADA systems in these industries will continue to grow, but at a slower pace. The level of activity in the worldwide SCADA market for these industries will be determined by a number of factors, including the level of energy exploration and subsequent delivery requirements, the rate of development of water infrastructure in developing regions, the results of ongoing geopolitical and environmental issues, and the emergence of cost effective technologies.

The majority of top SCADA suppliers offer a full suite of products and services, including RTUs and associated controllers. In addition to systems hardware, a full service offer also includes software applications, a complete line of communications technologies, and systems integration. These suppliers are also moving toward increased use of intelligent field devices, smart RTUs, and increased use of third party network infrastructure.

Electric Power Industry. According to the second study [24], the SCADA market in this industry will also continue to grow and is dominated by global suppliers. The emergence of large, global convergent energy and utility companies has enhanced the position of larger suppliers to match the end users' needs on a global basis. Alliances have become an important strategic consideration for all industry players, both large and small.

Suppliers are offering more advanced application software, business management tools, and the use of third party network infrastructures. SCADA is

moving towards knowledge management and is serving a more diverse range of client groups. This is becoming an important decision factor for users requiring enterprise and business integration. The functionality requirement is being expanded to include issues that were never considered before, such as energy trading.

2.3 Components of SCADA Systems

A traditional SCADA system consists of four main components: field instrumentation, remote terminal units (RTUs), master terminal unit (MTU), and communications. Figure 2.2 shows a typical SCADA system architecture.

2.3.1 Field Instrumentation

Field instrumentation refers to the sensors and actuators that are directly interfaced to the plant or equipment being controlled and monitored by the SCADA system. They are usually not considered part of the SCADA system itself but are an integral part of the overall control scheme. These field instruments convert physical parameters (fluid flow, velocity, level, etc.) to electrical signals (voltage or current) readable by the RTU equipment. Signals can either be analog (continuous range) or digital (discrete values).

Some of the industry standard analog instrument outputs of these sensors are 0 to 5 volts, 0 to 10 volts, 4 to 20 mA and 0 to 20 mA. The voltage outputs are used when the sensors are installed near the RTU. The current outputs are used when the sensors are located far from the RTU. Digital outputs are used to differentiate the discrete status of the equipment. Usually 1 is used

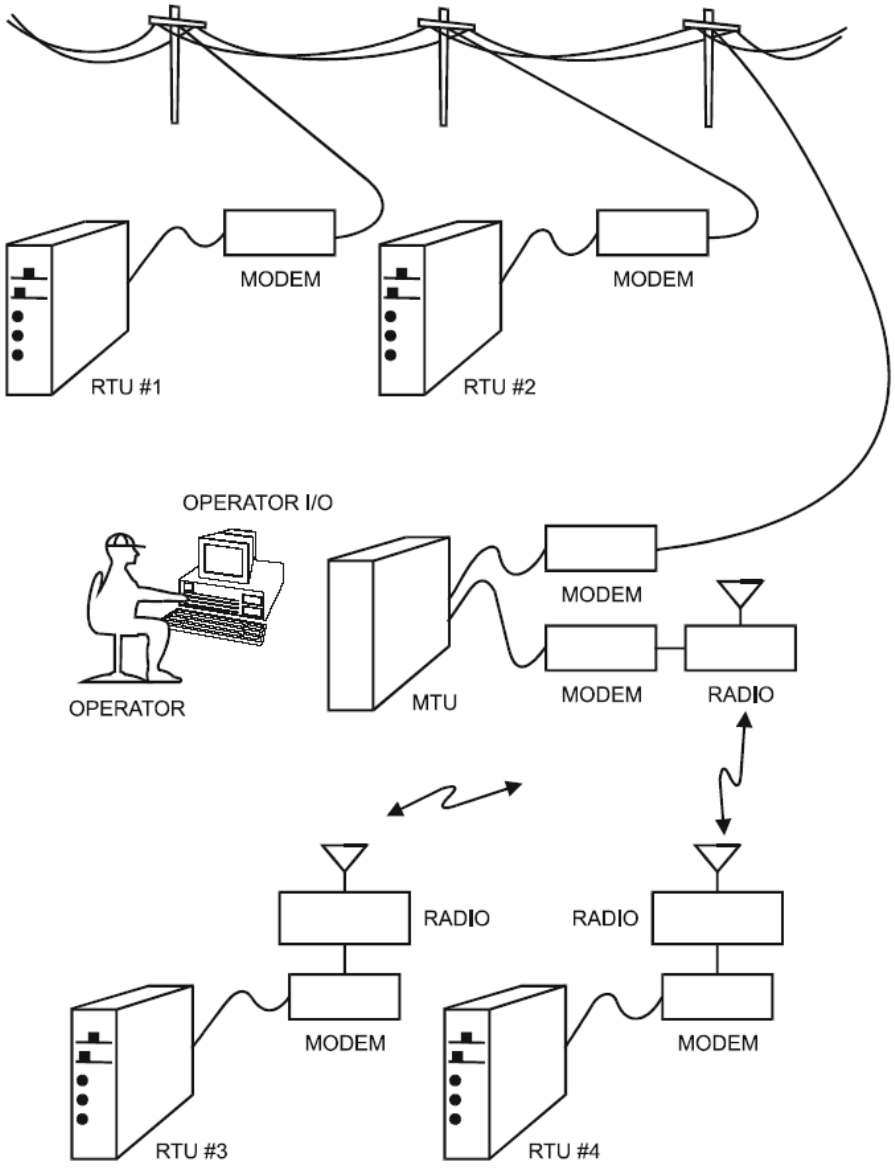


Figure 2.2: A typical SCADA system architecture

to mean ON and 0 for OFF status, or it could be 1 for FULL and 0 for EMPTY, and so on.

Actuators are used to turn on or turn off certain equipment (such as pumps) or open and close devices (such as valves). The digital and analog instrument inputs are used for control. For example, digital inputs can be used to turn on and off modules on equipment, while analog inputs are used to control the speed of a motor or the position of a motorized valve.

2.3.2 Remote Terminal Unit (RTU)

The Remote Terminal Unit (RTU) is a controller that is interfaced to the field instrumentation which is connected to the equipment being monitored and controlled. RTUs were initially developed as electronic black boxes whose flexibility of function was dependent on adding or removing hardware components. Customer demands led to increasing flexibility with programs written and burned into EPROM's. Recent technological advancements enabled even more flexible solutions by using computer-based RTUs. The so-called smart RTUs can perform the functions of PID controllers, fluid meter totalizers, and programmable logic controllers (PLC's).

RTUs are usually available in two types, namely, the single board and the modular unit. The single board provides a fixed number of input/output (I/O) interfaces. It is cheaper but does not offer easy expandability to a more sophisticated system. The modular type is an expandable remote station and more expensive than the single board unit. Usually a backplane is used to connect the modules. Any I/O or communication modules needed for future

expansion may be easily plugged in on the backplane.

PLCs are also being used with SCADA systems instead of RTUs due to their flexibility and programmability. However, the RTUs are usually designed to have very good radio interfacing since they are installed in difficult locations where communications are not readily available. Earlier PLCs did not have serial communication ports for interfacing to radio. But nowadays, PLCs have extensive communication features and a wide support for popular radio units being used for SCADA system.

2.3.3 Master Terminal Unit (MTU)

The Master Terminal Unit (MTU) is the master station or host computer which acts as the centralized controller for the SCADA system. The MTU of modern SCADA systems is always based on a computer. It could be a single computer configuration or it can be a network of computer workstations. It will also have auxiliary devices such as printers, loggers, and backup memories. The MTU is responsible for gathering field data from the RTUs and processing the information to generate the necessary control actions. The MTU is typically scheduled to request updates from the RTUs at fixed intervals.

The operator interface with the MTU is accomplished through the human-machine interface (HMI) software. The HMI provides a graphical representation of the process and its current status. The input value reading from each I/O point is displayed within its corresponding mimic diagram. Setup parameters such as trip values, limits, etc. are entered through the HMI and downloaded to the corresponding remote units for updating their operating parameters. The

HMI also has a separate window for alarm messages. The alarm window can display the alarm tag name, description, value, trip point value, time, date and other pertinent information.

Historical trend graphs can be viewed or printed at a later time. Generation of management reports can also be scheduled on for a specific time of day, on a periodic basis, upon operator request, or event initiated alarms. Access to the program is permitted only to qualified operators. Each user is given a password and a privilege level to access only particular areas of the program. All actions taken by the users are logged on a file for later review.

In many applications, the MTU is required to send accounting or management information to other computers or financial systems within the company. The MTU may also receive information from other systems or application programs which perform as higher level supervisory control. The connections between the MTU and the other systems may be accomplished via dedicated communication links or local area networks (LAN). The data exchange is established via standard or customized data communication protocols and interfaces.

2.3.4 Communications

Communications is the spine of SCADA technology. In order for the central MTU to communicate with the RTUs that are located at the distant locations or with the various computers and systems that are located within the corporate network, a communication link must exist to transfer data from one location to another. There are two common types of communication media:

wireline communications (electrical cable or optical fiber cable) and wireless communications (radio frequency). In either case, a modem or some other form of LAN technology is utilized. In most cases, a combination of more than one media is used.

Direct cable connections are usually not practical for large systems covering wide geographical areas. Therefore, SCADA systems typically use telephone lines which are either owned by the company or leased from the telephone utility. Leased lines can be used for systems requiring continuous online connection with the remote stations. Dial-up lines can be used on systems requiring updates at regular intervals. In critical applications where direct cable connections are required, optical fiber technology is becoming more affordable and is providing higher data rates and increased security.

For remote sites that are usually not accessible by telephone lines, the use of radio frequency (RF) communication offers an economical solution. Radio modems are used to connect the remote RTUs to the central MTU. Online operation can also be implemented using the radio system. For locations wherein a direct radio link cannot be established, a radio repeater is used to link these sites. In addition, satellite communications are becoming more common as services become more affordable. This alternative is particularly useful for remote sites which are located in rough landscapes or areas where direct line-of-sight could not be achieved for radio communication.

The type of communication scheme will determine the reliability and performance of the SCADA system. There are two modes of communication available, namely the polling system and the exception reporting system.

Polling Mode. In the polling (or master/slave) mode, the master is in total control of the communications. The master makes a regular polling of data (i.e. sends and receives data) to each slave in sequence. The slave responds to the master only when it receivers a request. This is called the half-duplex method. Each slave unit will have its own unique address to allow correct identification. If a slave does not respond for a predetermined period of time, the master retries to poll it for a number of times before continuing to poll the next slave unit.

The process of data gathering in polling mode is fairly simple, no collision can occur on the network, and any link failure can easily be detected. However, the disadvantages of polling systems include large waiting time with increased number of slaves, communication between slaves have to pass through the master with added complexity, and interrupt requests from a slave requiring immediate action cannot be handled.

Although polling systems are very common with wireline networks, polling gives poor performance over radio. For maximum speed with a polling system, the full radio frequency has to be utilized to ensure that each individual unit is polled as often as possible, and the radio frequencies may need to be licensed. In practice, a compromise has to be made between how often the system will poll and the amount of radio channel which is actually available.

Exception Reporting Mode. In the exception reporting mode, a unit transmits a message only when it detects a significant change in the process or when it exceeds a certain limit. The system is designed with error detection and recovery mechanisms to handle data collisions. Before any unit transmits, it must first check if any other unit is transmitting. If another unit is transmitting,

some form of random delay time is required before it tries again. Excessive collisions result to erratic system operation and possible system failure. To cope with this, if after several attempts, the slave still fails to transmit a message to the master, it waits until polled by the master.

The exception reporting system reduces unnecessary transfer of data as experienced in polling systems, which is useful for wireless radio communication. It also allows quick detection of urgent status information and allows slave-to-slave communication. However, the disadvantage of this system is that data collisions may cause delays in the communication and the master may only detect a link failure after a period of time. The operator may not have any indication of the failure unless he initiates some action (refresh screen) in order to see the latest process values.

System Configuration. Typically, a combination of polling and exception reporting systems are used for optimum performance. The polling is infrequent and is intended to check the integrity of the communications and the integrity of output values, while exception reporting is used for gathering the input readings. There are two typical network configurations for the overall system: (1) point-to-point system, and (2) point-to-multipoint system.

The point-to-point (or distributed) configuration involves data exchange between two stations at an instance of time. One station acts as the master and the other as the slave. An example is a setup of two RTUs: one for a reservoir or tank and the other for a water pump at a different location. Whenever the tank is nearly empty, the RTU at the tank will send an EMPTY command to the other RTU. Upon receiving this command, the RTU at the water pump will start

pumping water to the tank. When the tank is full, the tank's RTU will send a FULL command to the pump's RTU to stop the motor.

The point-to-multipoint (or client/server) configuration is where one device is designated as the master unit and the remaining ones as the slave units. The master is usually the main host and is located at the control room, while the slaves are the remote units at the sites. Each slave is assigned a unique address or identification number. This configuration is suitable for small stand-alone systems, but as the individual systems become interconnected, the point-to-point system allows greater flexibility and reliability.

2.4 Evolution of SCADA Systems

The main function of a SCADA system is to collect data from various remote sites and make it available in a central location for supervisory control and remote monitoring. Communications play a major role in SCADA systems development and advancement. Recent contributions in this area have been focused on two major parts: communication with the field-level devices, and communication with the high-level supervisory computers.

2.4.1 Field-Level Communications

This has been traditionally based on serial communication. But this is no longer adequate as modern industrial networks are built on Fieldbus and Ethernet networking technologies that leverage the more intelligent devices, allowing improved access to real-time measurements and instrument diagnostics. Also, as a SCADA system grows and operations cover larger areas,

diverse systems may have to integrate into a single contiguous system. The use of interoperable standards-based technologies becomes more important.

Standards make it possible to integrate equipment from different suppliers and bring the data all the way into the business environment. Some of the new standards being implemented in SCADA systems are Fieldbus technology, industrial Ethernet, and data communication protocols such as OLE for process control (OPC). Most of these new architectures are the result of close cooperation between SCADA system vendors and different oil and gas companies around the world, fusing their know-how with existing knowledge of Fieldbus and industrial Ethernet

Fieldbus digital protocols allow field-level devices such as transmitters, valve positioners, and remote I/O for discrete devices to network with each other rather than the 4-20 mA analog signals. Eliminating 4–20 mA removes the errors resulting from digital-to-analog-to-digital conversions in transmitters as well as in old RTUs. Fieldbus linking devices and flow computers take the place of remote terminal units (RTUs) from the past. Ethernet provides the media of choice for wired automation backbones, replacing a whole range of mainly proprietary protocols. Ethernet over radio, or wireless Ethernet, is also starting to find its way into industrial automation after its successful implementation in the business applications.

2.4.2 High-Level Communications

For integration with the higher-level supervisory computers, Ethernet and the Internet Protocol (IP) suite are gaining popularity in the industry. At the host

level in the central control room, Ethernet brings the data to an OPC server that provides the data to workstations for operation, engineering, and maintenance. The operations software may have a relational database that stores all data also available to other applications. The maintenance station runs the online plant asset management software. The database applications are usually made compatible with the Web, so that information can be dispensed to the corporate intranet or the Internet.

2.5 Cost of Implementing a SCADA System

The cost for implementing a SCADA system will vary depending on the size of the system and the complexity of its operation. A simple irrigation system with only one operator will not cost as much as a large electric transmission and generation system. The economic and business factors that influence a project evaluation will also vary depending on the specific process at hand. Typically this will require some form of a cost/benefit analysis. These costs and benefits have to be quantified before deciding what type of SCADA system to select. The costs may vary from area to area, but in general, they can be classified into four categories: capital costs, training costs, maintenance costs, and operating costs.

Capital costs normally include the engineering and installation labor, the services of technical specialists, warehousing, and transportation, in addition to the costs for the hardware and software. They are usually significant but easy to quantify. Capital costs may also include, as part of a project cycle, the removal of existing equipment to make room for new equipment, certain refurbishment of existing equipment or buildings, and custom-designed computer software.

Training costs are frequently neglected during the economic cycle. It is true that training costs are not very high compared to capital cost, but they do exist. There are costs for the time lost while operators are learning the new system, there are costs for maintenance technicians to get trained on the new equipment, and there are costs for acquiring the instructor and training material.

Maintenance costs involve the preventive and corrective maintenance of the MTU, the RTUs, the communication equipment, in addition to the calibration and repair of field instrumentation. The wire terminations that join all of these components into one system are a major source of maintenance cost. Technical support costs may include routine software upgrades.

Operating costs involve the cost of manpower needed to operate the system, which will depend on the size and complexity of the system as well as its operating philosophy. The operating costs also include energy consumption costs, consumable items costs (e.g. printer paper, backup media), and leased line telecommunications costs.

2.6 Disadvantages/Implementation Concerns

The concerns that are associated with the implementation of a traditional SCADA system can be summarized in the following:

Proprietary Vendor-Specific Solutions. Perhaps the biggest concern for the end-user is getting locked into a proprietary vendor solution. This can only be solved with the introduction of standards. Although there are efforts to develop standards in the SCADA system industry, there is still a lack of

consistent adherence to these standards. This means that true interoperability between different equipment is virtually impossible to achieve without special drivers and interfaces.

Non-Standard Application Layer. Although common networking standards like Ethernet and TCP/IP are becoming the de facto standards in industrial automation, the situation is not so simple. Figure 2.3 shows the difference between typical industrial networks and the Internet protocols (using the ISO/OSI reference model.) Each vendor of automation equipment that runs over Ethernet and TCP/IP has implemented its own application layer protocol. As a result, a standard application layer, common object model and universal device profiles do not exist. Ethernet and TCP/IP only cover the lower layer protocols. At the application layer, users are currently tied to proprietary solutions and are not able to benefit from the best-in-class and best-in-value options offered by an open market.

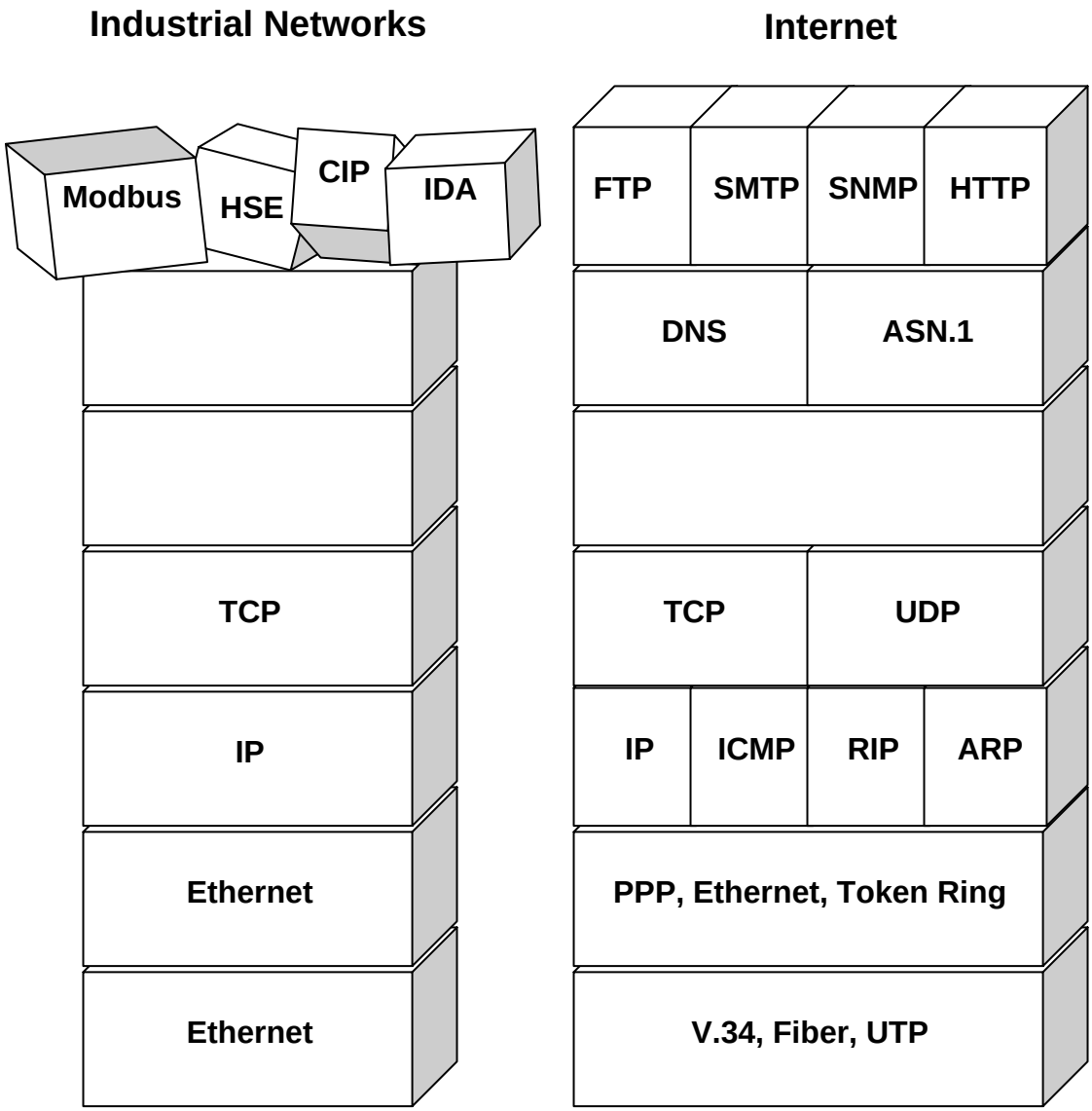


Figure 2.3: Typical industrial networks compared to the Internet

CHAPTER 3

INTERNET-BASED SCADA SYSTEMS

In this chapter, we provide an overview of Internet technologies that could be applied to SCADA systems, and how they are playing a major role in the evolution of these systems. We present some case studies from the oil and gas industry, and then we analyze the advantages and disadvantages of such implementations. This analysis, along with the next chapter, will set the stage for our research hypothesis which is to evaluate how Internet-based SCADA systems can be improved upon.

3.1 Internet Technologies Applicable to SCADA

3.1.1 Networking Technologies

The Internet is merely a connection of regional wide area networks. In order to analyze what networking technologies are used by the Internet and could be applied to SCADA systems, we will use the International Organization for Standardization's Open Systems Interconnect (ISO/OSI) reference model. This networking model consists of seven layers as shown in Figure 3.1. Below is a brief description of each layer.

Physical Layer. This is the lowest layer in the model. It defines the physical characteristics of the connection media which are used for transmitting

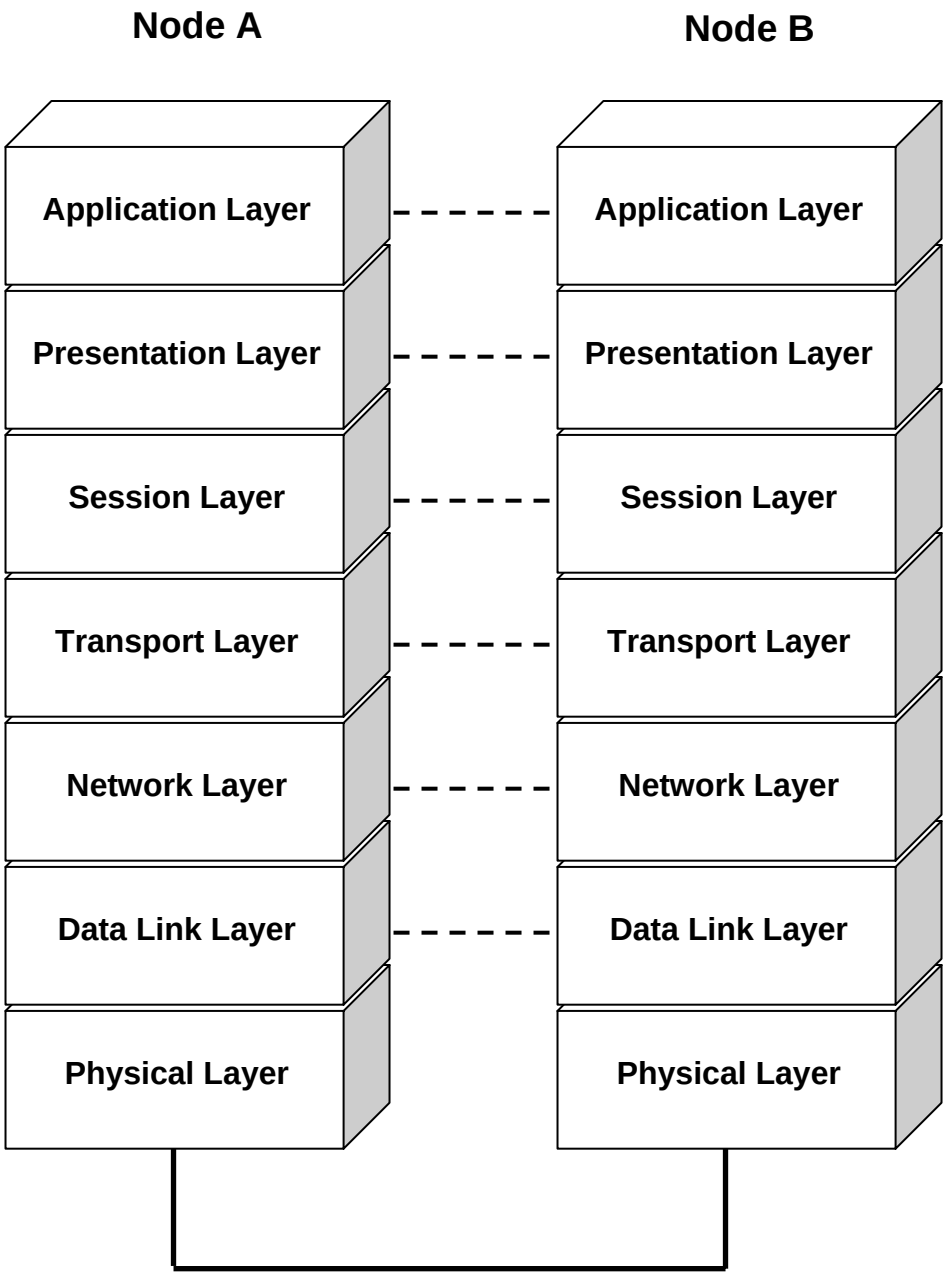


Figure 3.1: ISO/OSI network model

the electrical signals between different nodes on the network. The Internet usually encompasses a variety of physical media, both wired and wireless media. These can readily be utilized by a SCADA system for communication over large geographical distances.

Data Link Layer. This layer defines the rules for framing, converting electrical signals to data, error checking, physical addressing, and media access control. Examples of protocols used by the Internet are Point-to-Point Protocol (PPP), Ethernet, Token Ring, etc. Ethernet actually covers both physical and data link layers, and is becoming widely used in industrial automation. Industrial Ethernet can be utilized by a SCADA system for reliable and field proven networking.

Network Layer. This layer defines the way messages are routed through a complex network. The Internet Protocol (IP) is the standard method of network routing on the Internet. This is also becoming a de facto standard in industrial automation, and can be easily applied for SCADA systems communicating over the Internet.

Transport Layer. This layer defines the connection methods between two end nodes. The Internet Protocol (IP) Suite actually includes, in addition to the IP protocol, the Transport Control Protocol (TCP), and the User Datagram Protocol (UDP). TCP/IP is becoming widely used in industrial automation and can be applied for Internet-based SCADA systems.

Session layer. This layer defines what messages are needed to complete a session. This layer is a vague concept and is usually skipped or built into the upper layers for industrial networks. It is sometimes used in Novel and

Microsoft Windows networks.

Presentation Layer. This layer defines the data format conversion, encryption, and security. This is also skipped or built into the upper layer for most industrial networks. On the Internet, this layer is used by some conversion protocols, such as the Domain Name Servers (DNS).

Application Layer. This layer defines the rules for network applications. Internet applications that reside on this layer include the File Transfer Protocol (FTP), Hyper Text Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), Simple Network Management Protocol (SNMP), Telnet, etc. An Internet-based SCADA system could utilize some of these protocols, perhaps the most important of which is HTTP which enables web browsers to access web servers.

3.1.2 Web Technologies

The Internet connects people together, people with machines, and machines with machines, over a global network called the World Wide Web. In this section, we provide an overview of basic web technologies. In the next chapter we focus on two specific web technologies and explain how they could benefit SCADA systems.

Web Servers. A web server is simply a computer designated to send out data in response to a request. Basic web servers respond to requests over the Internet for specific web pages (e.g. HTML text) and web objects (e.g. graphic files like JPEGs). Specialized web servers can be used to do more focused functions. For example, e-commerce servers provide safe and secure economic transactions for doing business over the Internet; communication servers

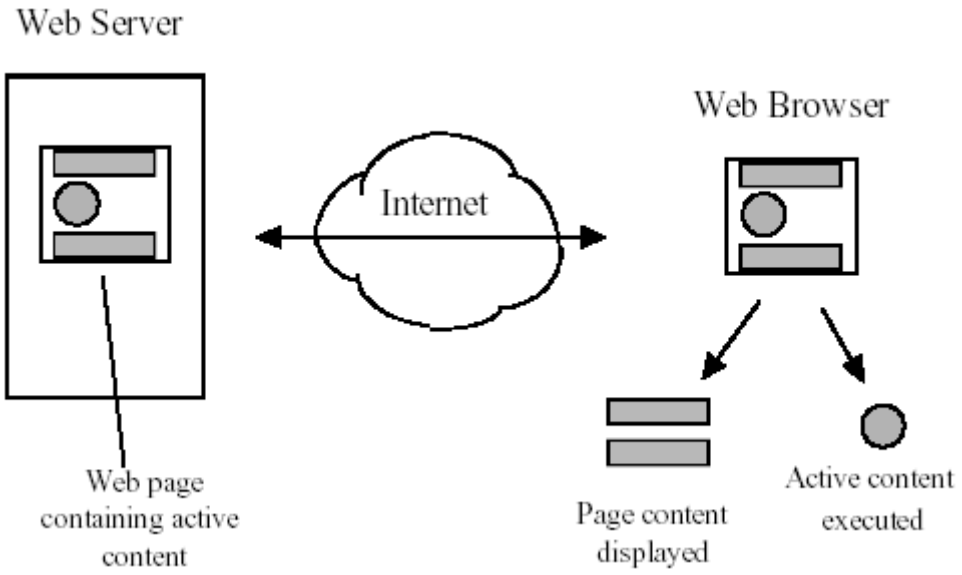


Figure 3.2: Web servers and web browsers

provide ways for people to talk to each other via the Internet whether through text (chat), e-mail, or voice (phone). SCADA systems could utilize this concept by implementing the Master Terminal Unit (MTU) as a web server, or even each of the Remote Terminal Units (RTUs) as a web server. This allows the SCADA operator to access process information from the MTU and RTUs via standard web technologies.

Web Clients (Browsers.) Servers send requested information to clients or browser software. There are several browser software available, such as Microsoft's Internet Explorer and Netscape's Navigator, but all essentially do the same thing: request data from a server and display it on a page on the client computer. Some browsers offer more diverse functions than others such as the ability to display and interpret special technologies. New features appear every day and browser manufacturers often release new versions of their browsers (updates). SCADA systems could utilize the browser technology to implement a unified human-machine interface (HMI).

Web clients can be divided into two main types based on their functionality. The client that uses the standard web browser to access web applications is often called a "thin client." The client that requires additional software applications to be downloaded and installed on the client machine before accessing a specific web application is called a "thick client." Thick clients usually take longer time for the initial setup, but provide maximum functionality to the user. On the other hand, thin clients only require minimum or no initial setup, but on the expense of functionality and versatility. Table 1 below shows a comparison between the functionality of a stand-alone distributed network

TABLE 3.1: FUNCTIONAL COMPARISON BETWEEN WEB CLIENTS

	Stand-Alone Distributed Client	Thick Web Client	Thin Web Client
Client Software	Vendor-specific application	Vendor-specific application + Web browser	Web browser
Installation and Maintenance	Requires installation of the client software on each client machine	Requires installation of the client software (either manually or automatically from the web)	Does not require software installation on the client side
User Interface	Full graphical functionality + popup windows + dynamic data	Standard browser functionality + VB/Java scripting	Standard browser functionality
Performance	Generally fast	Generally fast	Slower than thick clients

application, and that of web clients.

Web Application Development. The Hyper Text Markup Language (HTML) has been the main language for developing web pages and linking them over a wide area network. However, in order to develop specialized web applications, HTML is usually limited in terms of functionality. Hence, to compensate for this, Java applets or other scripted code, like JavaScript and VBScript, are embedded into the HTML documents to enhance their functionality. Also, a new markup language called XML (Extensible Markup Language) has been introduced to improve upon the limitations of HTML.

The World Wide Web Consortium (W3C) develops and maintains the specifications for HTML, XML, and many other technologies, such as the Scalable Vector Graphics (SVG) specification used for handling graphical user interfaces. The primary purpose of W3C includes specifying and promoting standards for technology and software that programmers use with the World Wide Web. The W3C consists of many different companies, but the products that they support do not tie to any specific company and are freely available for any individual or company to use or implement.

On the other hand, the Java programming language has been introduced by Sun Microsystems to enable programmers to develop web applications that can be downloaded to distributed machines on the Internet. Java programs are portable in that they can be written and compiled on one computer and loaded and run on another regardless of the hardware or operating system of these computers. Java can be used to develop distributed applications in a stand-alone client/server fashion, or it can be used to develop Java servlets and

applets. The Java servlet runs on the server side, while the Java applet is embedded into the client web pages.

In the next chapter, we focus more on the Java and XML technologies and discuss how they are used in the development of new generation of web applications, namely in the web services. We will then investigate how SCADA systems can benefit from these new web technologies.

Domain Names and URLs. Each page on the web server has a unique Universal Resource Locator (URL) address. The URL consists of the domain name, followed by the actual file (directory) structure on the server. An Internet-based SCADA system could easily use this addressing scheme to access the web-based MTU and RTUs. Alternatively, the network IP address could be used in the URL instead of the domain name, which makes the addressing task more familiar and analogous to the traditional SCADA system's addressing and naming conventions.

The most common protocol used for accessing these URLs is the Hyper Text Transfer Protocol (HTTP) which is an application layer protocol that defines the way messages are exchanged between a client and web server. HTTP usually establishes the connection between the client and web server using TCP port 80. This port is often left open on most corporate firewalls to enable Internet access to employees. An Internet-based SCADA system could use HTTP for remote communications, enabling it to communicate with applications across firewalls. This is an important differentiation between HTTP and other methods of communication, such as API or RPC sockets, which are often blocked by the firewalls.

3.2 Definition of Internet-Based SCADA

The term Internet-based SCADA has been used loosely in the industry to describe a SCADA system that applies one or more of the Internet technologies. This may include communication technologies, software programming technologies, and web browser technologies. The key goal of implementing Internet-based SCADA is to utilize internationally accepted standards and technologies to achieve the monitoring and control functions that a traditional SCADA system provides but with a lower cost. This will usually result in better interoperability between different system components, easier dissemination of information to various applications and external systems, and unification of the human-machine interface (HMI) through the standard web browser.

Interoperability, or the ability of system components to interact with each other successfully when connected in a specified way, is achieved by using a set of common protocols and standards. An internet-based SCADA system is usually based on one of the common networking standards such as Ethernet, Token Ring, and ATM (which correspond to layers 1 and 2 in the OSI network model), in addition to the Internet protocols suite which includes (among others) the Internet Protocol (IP), the Transport Control Protocol (TCP), and the User Datagram Protocol (UDP) (OSI layers 3 and 4).

Dissemination of process information across the plant or organization is also achieved by using a set of common applications and open interfaces. An Internet-based SCADA system typically uses web technologies, where a web server stores the information and a web browser reads or writes the information. The data presentation is usually done using the Hyper Text Markup Language

(HTML). However, HTML is limited in terms of functionality, and thus the Extensible Markup Language (XML) was created to overcome the shortcomings of HTML and to be used for sending data between the server and browser (or the server and another server). XML is slowly finding its way into industrial automation and is replacing many of the proprietary vendor protocols.

The Internet-based SCADA system may also include embedded smart devices (such as drives, motors, servos, actuators, gauges, pumps, flow meters, etc.) which have built-in embedded web servers that can transfer data from the plant floor all the way up to the enterprise web browsers, thus allowing remote control, diagnostics, asset management, and supply chain management. The web browser also acts as a unified human-machine interface (HMI) which combines the graphical operator interface, data acquisition, and alarming functions, with the capability of real-time enterprise integration and external data communication. The web browser may require additional plug-ins to provide HMI functionality close to that of traditional SCADA systems.

3.3 Case Studies from Oil & Gas Industry

3.3.1 British Petroleum

British Petroleum (BP) [25] installed a wireless Internet-based SCADA system to monitor and control its remote gas well sites at BP Canada Ltd. The new SCADA system is similar to a traditional SCADA system in that the same multivariable flow transmitters (field instrumentation) are used to communicate with the local RTUs. The RTU receives process information from the transmitter and performs gas flow calculations, stores hourly production data, controls

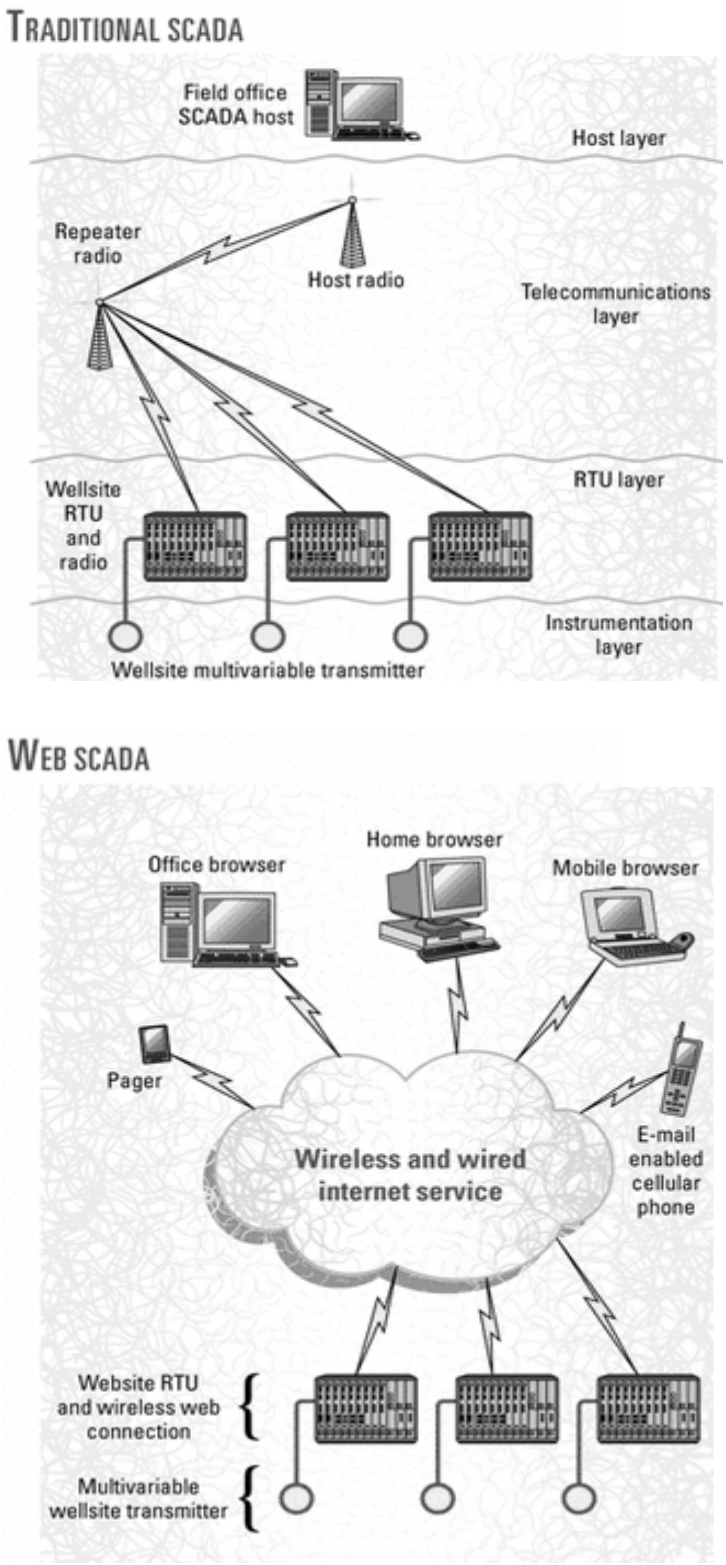


Figure 3.3: BP's system implements each RTU as a web server

input-output (I/O) points, and performs ladder logic. However, instead of acting as a slave to an MTU host, each RTU assumes the role of a web server.

Each RTU has a unique web address which is accessible from any web browser. The communication is established through the wireless network which exists in almost all gas producing areas. The wireless web is usually a digital component of the cellular telephone spectrum with the internet service provider (ISP) typically being the cellular phone company. The coverage generally overlaps the same territory where voice coverage exists.

For BP, this was a key to the project's success. Its well sites, although in difficult-to-reach terrain, all had good wireless web coverage. BP's local ISP-cellular phone service provider, Telus Mobility, Calgary, uses cellular digital packet data technology to deliver the wireless internet service. The implemented web-based solution provides many of the same features as a traditional SCADA system including real-time view of data, alarm notification, data logging, and remote configuration

The operator can monitor and control the well site from the field office, home, or vehicle. Handheld PCs, palm-size organizers, or laptop computers can all connect to the wireless web, allowing the operator more mobility and productivity. E-mail is one way operators can be alerted to a problem such as no-flow alarms. When the flow rate drops below the preconfigured low-flow threshold, the RTU will generate an alarm e-mail that informs the operator of the abnormal condition.

Data management may become a problem for operators with more than one or two well sites equipped with a web-based RTU. One way to handle these

data is by creating a small Visual Basic application that works with Microsoft Outlook to manage the incoming daily e-mails and store the data in a more usable format. The Visual Basic application then continually scans the inbox looking for messages generated from one of the well site RTUs.

3.3.2 Shell

Shell [26] developed an end-to-end Internet-based SCADA system (called eSCADA) on 12 offshore production platforms. Each platform has several vmBus transmitters (supplied by vMonitor) that are interfaced to various electronic transmitters, instruments, and thermo elements, which measure temperature, pressure, density and other parameters. The various transmitters create a wireless mesh on the remote platforms which transmits the data to a central hub. The hub then sends the data to the main platform, at a distance ranging from 3 to 18 km. Some units have two-way communications for remote control.

The system was first implemented during a field trial in the Urdaneta West field, in the western part of Lake Maracaibo, Venezuela. The test involved interfacing the wireless vmBus unit to three analog inputs (4-20 mA) from an existing field transmitter measuring temperature, pressure, and density. The data is sent from the offshore platform back to the main platform via wireless communications and without an RTU. The vmBus unit operates at 900 MHz and has four analog and four digital inputs. The vmBus transceiver is interfaced via RS232 connection to a long range vmBus radio, which then transmits the data to the main platform.

At the main platform, a web server hosting vMonitor's TotalAccess

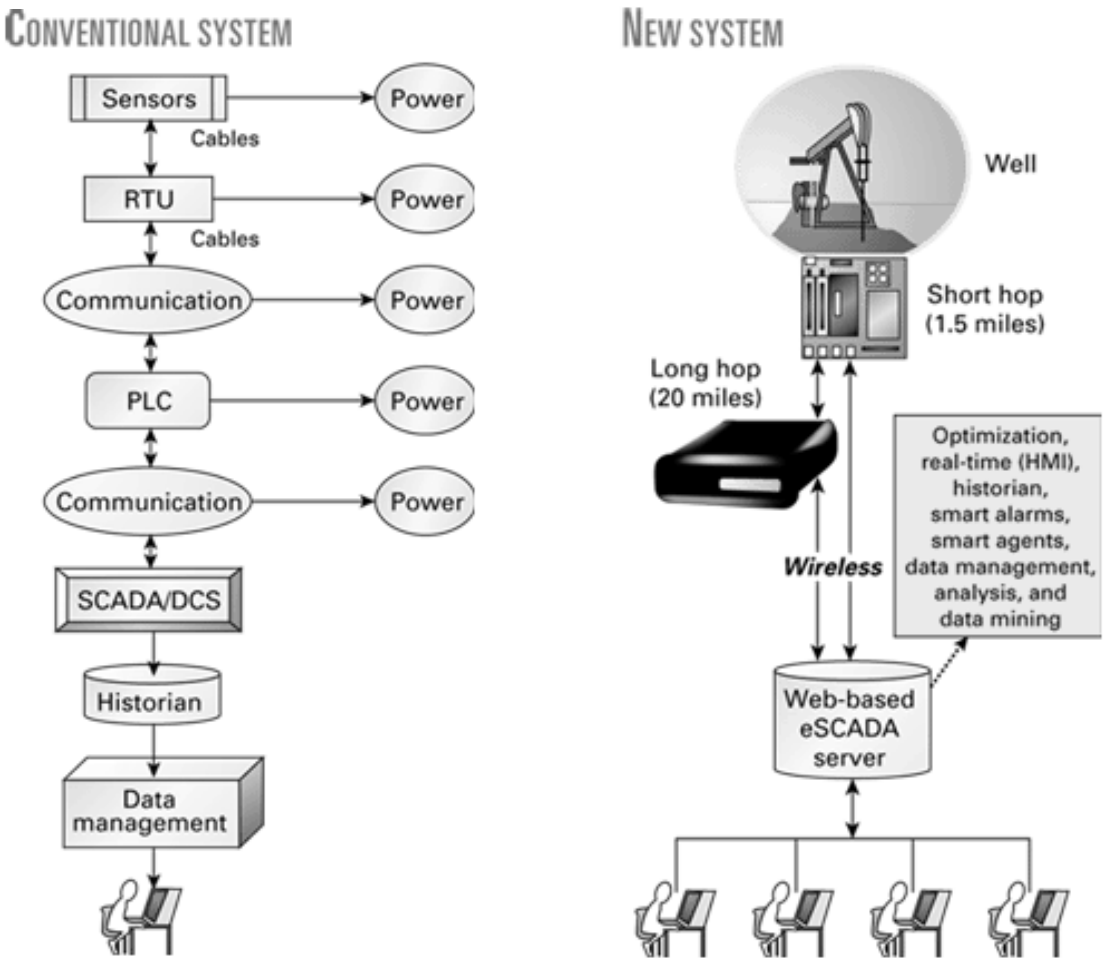


Figure 3.4: Shell's system implements the MTU as a web server

software provides web-based, real-time monitoring HMI, smart alarms, reports, and historian. The data is also interfaced with a Foxboro's Distributed Control System (DCS) and OSIsoft's Plant Information (PI) system. Operators at the main platform are able to monitor the pressure, temperature, and density readings from the offshore platform. The TotalAcess software also obtains readings on a laptop server on a boat en route to the main platform without any interference or interruption.

3.3.3 ChevronTexaco

ChevronTexaco Refining Company [27] has developed a portal website on the corporate intranet (called CommonView) to bring integrated information to the desktop via the web to enable better decision making, information sharing and best practice support. The portal web site is targeted at a specific audience providing content aggregation and delivery of relevant information. Information in the right hands at the right time is key to achieving operational excellence and organizational capability.

The CommonView portal (implemented by IndX Software) uses Java and XML to create a common data visualization tool for users across and within the refineries. CommonView aggregates the information from multiple data sources including process history data, maintenance management, procedures, turnovers, safety and environmental data, reliability system, document management, production accounting, handhelds, 3rd party applications and services, etc.

The CommonView Portal has been implemented at most ChevronTexaco

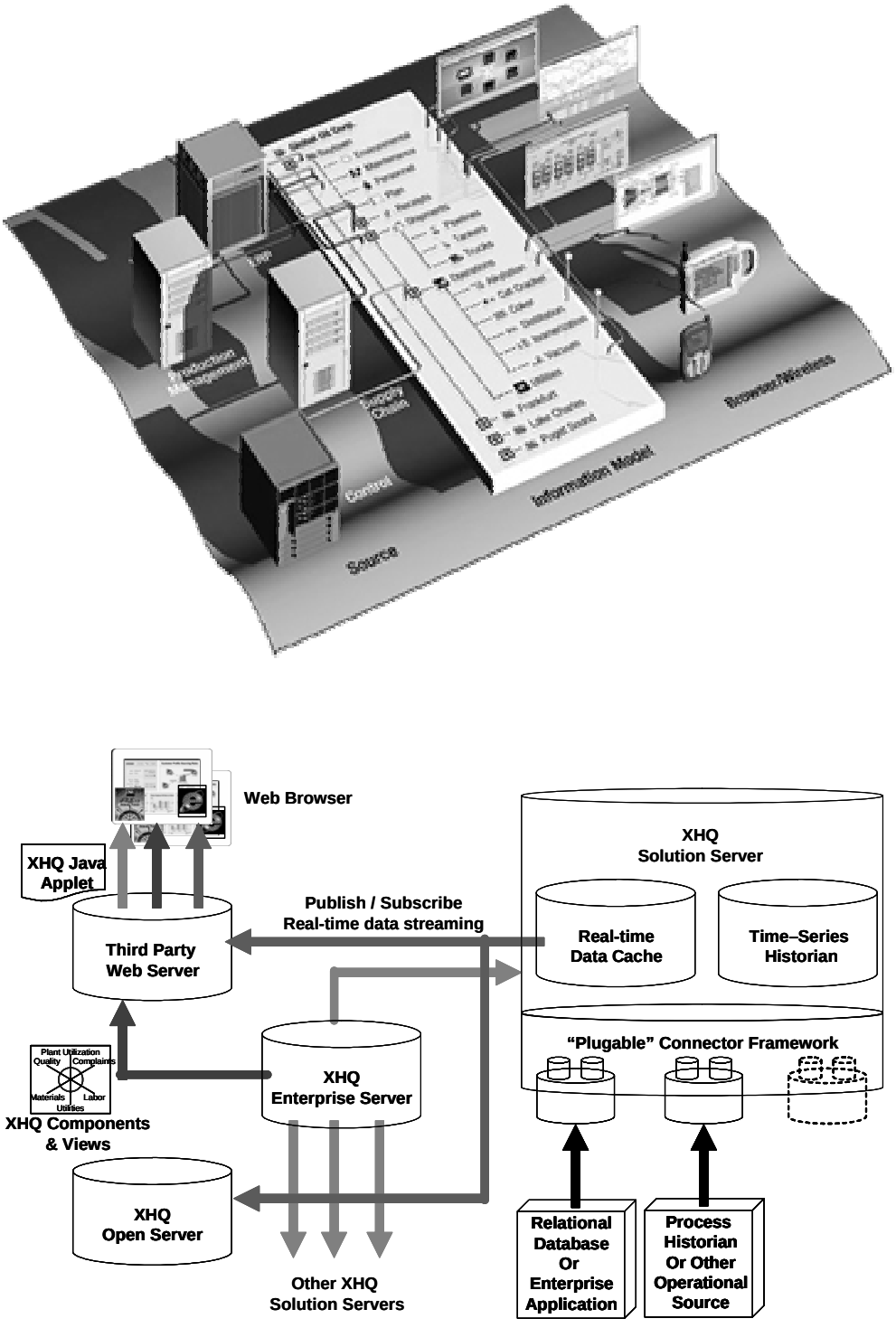


Figure 3.5: ChevronTexaco's web portal uses Java and XML

refineries including El Segundo, Richmond, El Paso, Hawaii, and Pembroke Refineries. The primary users are operators, engineers and operations management. Secondary users are maintenance, environmental and refining management. Cross-refining users are refining management, marketing, and logistics.

3.3.4 Saudi Aramco

Saudi Aramco [28] has been evaluating several web portal solutions that can consolidate data from various process systems and organize them into a web-based solution for supporting decision making and process management. The Intranet-based systems are usually targeted for management information and decision support mainly, with no control being performed on the remote sites.

The system handles process data from different sources including process control systems (DCS, PLC, SCADA), process history data (Plant Information - PI Systems), enterprise resource planning system (SAP), maintenance management, procedures, turnovers, document management, production accounting, third party applications and services, etc. Presentation on the web includes a graphical user interface and integrated office applications functionality.

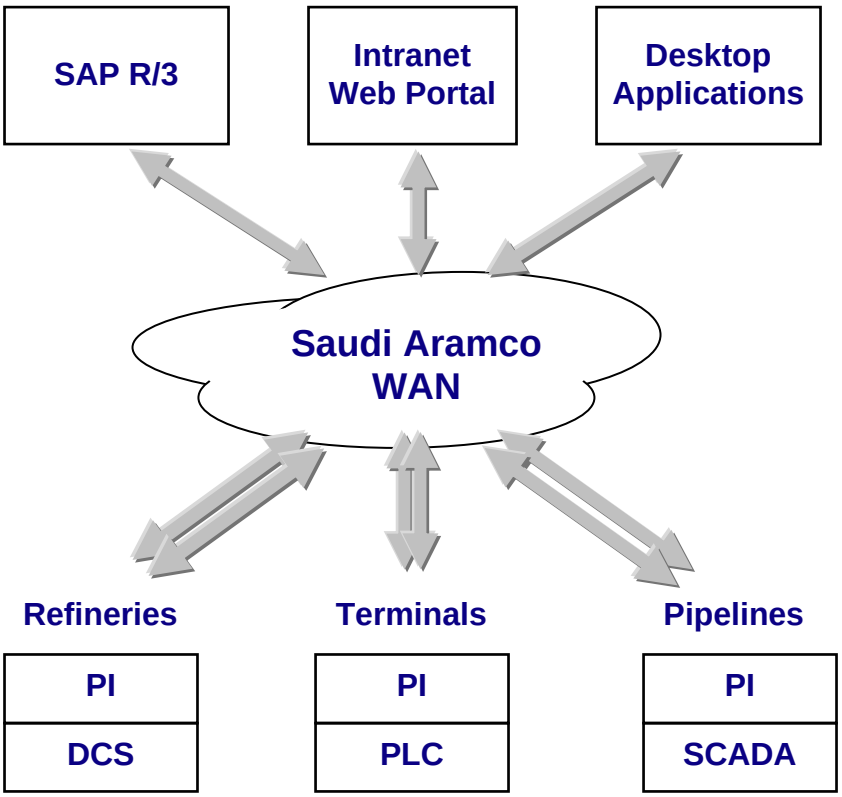


Figure 3.6: Saudi Aramco's integrated systems

3.4 Advantages of Internet-Based SCADA

The main advantages of Internet-based SCADA systems compared to the traditional systems can be summarized in the following:

Open Standards. The main advantage of implementing an Internet-based SCADA system is that it is a standards-based system which leverages the existing computer and communications technologies to achieve optimum system functionality with minimum cost. An Internet-based SCADA system utilizes one of the existing application layer protocols to provide the openness required for a multi-vendor environment, rather than getting locked into one of the proprietary network solutions.

Interoperability. The fact that an Internet-based SCADA system is based on standards shall increase interoperability between the various system components regardless of what vendor is supplying the individual components. Interoperability benefits shall cover not only the operation side, but also the maintenance functions and instruments diagnostics that can be obtained by the system.

Simple and Familiar. Internet-based SCADA systems are generally easy to learn and use because the browser navigation tools are familiar to anyone who has surfed the Internet. This translates to less training requirements and faster learning curves.

Expandability. By using standard web server/web browser technology, an Internet-based SCADA system can extend to other parts of the organization by using the intranet and web-based portals, enabling corporate executives and other business decision makers to aggregate and analyze data from multiple

plants worldwide, regardless of time zones. They can also quickly produce needed production reports, downtime tracking reports, batch reports, WIP reports, quality assurance reports, and so on.

Reduced Project Costs. The cost for implementing an Internet-based SCADA system will generally be lower than a traditional system. This is due to the fact that the Internet-based SCADA system uses standard hardware and software, requires lower training costs for operators and maintenance technicians, requires lower preventive and corrective maintenance costs, and possibly involves lower operating costs as the system architecture and operating philosophy is simpler.

For example, in the first case study that we presented (the BP implementation) [25], it was reported that the cost of implementing the new web-based SCADA system is lower than a traditional system, mainly because there is no need for a dedicated host computer or dedicated communications network. Figure 3.7 shows how the cost-per-wellsite remains constant regardless of how many wellsites are automated.

3.5 Disadvantages/Implementation Concerns

Although the concept seems appealing to many organizations, some important issues are still deterring them from full implementations. These issues need to be considered carefully as they have an impact on the system communications, openness, security, functionality, and performance.

Non-Deterministic Communications. Determinism is the ability to predict when information will be delivered. All networks provide some degree of

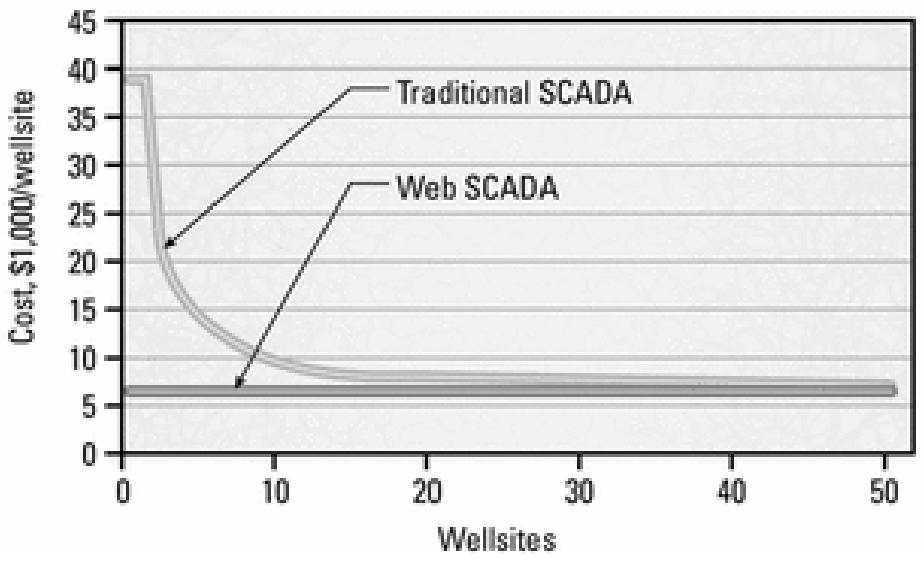


Figure 3.7: Cost comparison (from BP case study)

determinism. If a network only has two nodes — and the transfer of data between the nodes is restricted to avoid collisions — the network has absolute determinism. However, in realistic situations, the number of nodes on an Internet- or intranet-based system continually changes. In order to guarantee determinism, a network must provide scheduled bandwidth (or time slots) that is reserved for time-critical data transfer.

Security Concerns. There are still some concerns related to the risk of hackers or other intrusions affecting the performance and functionality of the process systems, either intentionally or by mistake. The normal practice is to protect these systems through network firewalls and password protection, which will reduce the risk of unwanted access, but cannot guarantee 100% protection. Also for real-time SCADA applications, a single level of security may not be sufficient. Several systems implement what can be called a point-by-point security. Each user is given access on a per-tag basis. This way, critical tags or field outputs can be given 'no access' for users.

Limited Browser Functionality. Current restrictions in terms of functionality come from the fact that web browsers are limited to HTML functionality, providing limited support for pop up windows, and dynamic allocation of data links or memory. To get suitable functionality close to a SCADA's, the browsers need a significant number of plug-ins to be loaded on the client, which has impacts on performance and maintenance. There is an assumption that no maintenance is involved on the client end. In most cases this is not true, as plug-ins need to be downloaded on the client. The amount of plug-ins may vary from system to system based on the requirements.

HMI Performance. Performance is critical in SCADA systems especially with the large number of human machine interface (HMI) graphics. Even with large bandwidth systems, the client must be responsible for most of the graphics handling and database update. Few vendors build their graphics using formats that retain the integrity of objects. The majority of attempts used to be by converting the real-time graphics into JPEG or GIF pictures, which loses the object elements. The W3C committee has selected the Scalable Vector Graphics (SVG) format as its standard which improves graphics handling. In addition, there are mechanisms on the server side that improve the performance of displays, e.g. by refreshing them. The W3C standard to do this is called Simply Object Access Protocol (SOAP). SOAP objects are based on XML and are used to access properties and execute methods across the web. This will be discussed further in the next chapter.

CHAPTER 4

JAVA AND XML

In this chapter, we discuss two specific web technologies, namely Java and XML. We explain why these two technologies were chosen for our research hypothesis, what each of them represents, and what benefits could we get from combining them together. We will then present some of their applications in the e-business world. This will be followed by discussion of their possible applications in the process automation world.

4.1 Why Java and XML?

Java is the most common programming language for the Internet [29]. The developers at Sun Microsystems intended it to be a platform-independent programming language so that it could run on a variety of machines under different operating systems. This was done at a time when the Internet was emerging and gaining popularity. Java allowed people working on different machines under different operating systems to download content and applications from the Internet, and run them locally on their machines. In other words, Java provided "portable code."

The Extensible Markup Language (XML) is a method for structuring and describing information [6]. It is a subset from the Standard Generalized Markup

Language (SGML), which is a specification laid down by the World Wide Web Consortium (W3C). SGML is a generalized language for structuring information. However, it is too complex to be used for most applications. XML is more geared toward creating one type of content. It is an efficient and effective way of storing and sharing information, making it possible for a wide range of applications to easily share data in a controlled and consistent manner. Therefore, XML provides "portable data."

Since Java programs can be created on one platform and executed on another, and since XML information can also be formatted on one platform and transmitted to another, combining both Java and XML together lead to dual portability of code and data. Wherever Java programs can run, they can also access XML information. This enables Java and XML information to interoperate efficiently and effectively on different platforms. [30, 31, 32, 33]

4.2 Overview of Java

4.2.1 Java Platform

The portability of Java is accomplished by using a Java Virtual Machine (JVM), shown in Figure 4.1. When Java source code is compiled, the result is not a standard executable, but a bytecode which can be interpreted by the JVM. The JVM converts the bytecode back into the appropriate machine instructions. Different JVMs are available for different operating systems and they all adhere to a strictly defined specification. The bytecode generated on any machine running any operating system can be interpreted by a JVM on any other machine.

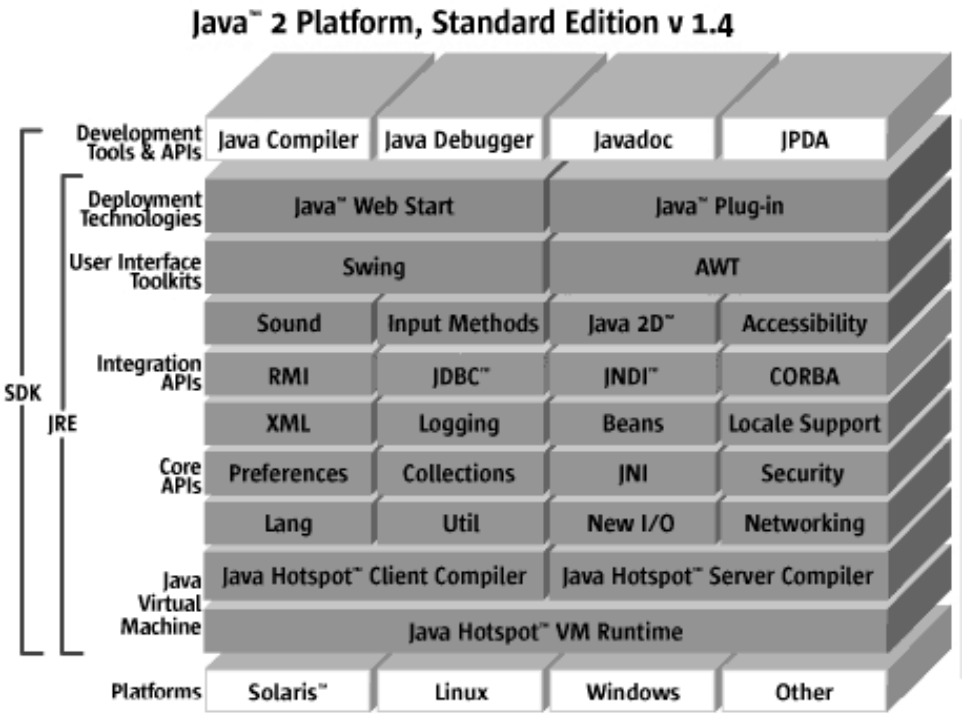


Figure 4.1: The Java Platform

4.2.2 Java Program Structure

The Java language organizes its programs in a hierarchical fashion, making them easier to read and understand. The fundamental unit of Java software is the class. A class is a description of a type of object, and includes a collection of data and the code that operates on the data. A typical Java program consists of hundreds of classes, some written specifically for that program, and many others selected from the extensive Java class libraries. Typically, the classes are grouped into packages. The categorization of classes into a package is done by their function.

All Java code appears inside of classes. Furthermore, all executable statements appear inside of methods, which reside in classes. No global variables or functions may appear outside of any class, as happens in C++. It is also not possible to write a single line of Java code the way a single line is written in Perl or JavaScript. Java's rigorous structural rules become very useful when developing large distributed applications.

4.2.3 Java Characteristics

Portable Code. The platform-independent nature of Java makes it possible for application code to be transmitted from one machine to another over the Internet regardless of the hardware platform or operating system used on those machines.

Simple and Familiar. Java is based on the familiar C and C++ programming languages and borrows much of basic syntax from those languages. The developers at Sun Microsystems wanted to make Java a simple

language by removing some of the complex elements of C and C++ and adding other important features missing from C and C++. The removal of programming elements was aimed partly at minimizing redundancy and overlapping features in C and C++.

Object-Oriented. Java was designed from the ground up as a fully object-oriented programming language. The object-oriented paradigm has become the model of choice for modern programming languages in that it supports the needs of client/server and distributed software. Objects have portability and persistence. They can be created in one location and sent over a network connection to another location to be used or stored for future use.

Robust. The Java compiler performs an extensive check for syntax-related errors and warnings, so problems can be identified before the program is deployed. Once the program is deployed, a sophisticated exception handling capability is built into the language to ensure that exceptions that would cause program termination can be dealt with.

Secure. The Java compiler and runtime have built-in features to prevent application programmers from writing malicious code as Java does not support publicly accessible pointers. Also, the Java class loader implements additional security procedures to prevent a remote class from spoofing (pretending to be) a local class. Additional security mechanisms are provided by the bytecode verifier and by the interfaces contained in the Java networking package.

Multithreaded. A Java program can start multiple threads of execution, each performing its own sequence of operations at the same time. The Java language provides the tools for threads to acquire and release locks in a manner

that minimizes deadlock conditions. Java also provides programming mechanisms to synchronize access to methods and develop functionality in a thread-safe manner.

Versatile and Expandable. The Java language is versatile in that it covers a wide range of programming disciplines, such as network programming, graphical user interface (GUI) development, and creating different applications using a single programming language. There is no need for integrating between different languages using Common Gateway Interface (CGI) scripts or other mechanisms. Also, the Java language is expandable in that new class libraries are constantly being made available for programmers to download and incorporate into their applications.

4.3 Overview of XML

A markup language uses tags embedded directly into a text file to describe the various parts of the text. It adds labels to bits of text, and based on the label, the outputs device can decide how best to process the content. A markup language does not worry about how the content will be formatted; instead, it is concerned with describing the content in an accurate and consistent manner.

The Hypertext Markup Language (HTML) is probably the most widely known markup language. HTML evolved to contain tags that are used solely for formatting the display of information. Because of this, it is considered to be limited when it comes to storing many different types of information or describing the structure of information. This is why another markup language is needed. XML was developed in an effort to focus more on the content of

information rather than on the formatting and displaying for that information. Therefore, XML separates structure from display, allowing the application to decide on how best to present the data.

4.3.1 The XML Document

An XML document is the file that contains the XML data. It can be broken down into two basic parts: the header, which gives the XML application the information it needs about how to handle the document; and the content, which is the XML data itself. Figure 4.2 shows a sample XML document.

Well-Formed Documents. The XML data is organized into elements and attributes. The root element is the highest-level element in the XML document, and must be the first opening tag and the last closing tag within the document. All other elements are enclosed within the root element. Attributes are the parameters that follow an element's opening declaration, and are usually used for single-valued data. If the XML document follows all the rules of correct XML syntax, it is said to be well-formed.

Valid Documents. Because XML is extensible and can represent data in many ways, constraining a document provides meaning to those various formats. If the XML document follows the constraints set upon it, it is said to be valid. There are two main standards for constraining XML, which are explained in the following section.

4.3.2 XML Constraints

XML documents are not very usable without an accompanying constraint.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE device SYSTEM "C:\Documents and Settings\
Ramadan\My Documents\XMLspy projects\device1.dtd">
<device>transmitter
   <id>
         <tag>TT-005</tag>
         <variable>temperature</variable>
         <manufacturer>XYZ</manufacturer>
         <model>ABC</model>
   </id>
   <measure>
         <unit>C</unit>
         <value>31.0</value>
         <quality>good</quality>
   </measure>
   <service>
         <settingdate>05.12.04</settingdate>
         <operator>Ramadan</operator>
         <precision>0.45</precision>
   </service>
</device>
```

Figure 4.2: Sample XML document

Without document constraints, it is impossible to tell what the data in a document means. There are two current standards for XML constraints: Document-Type Definitions (DTDs) and XML Schema.

Document-Type Definitions (DTDs.) The DTD, or sometimes called the vocabulary, is a file that defines how the data is structured in the XML document by specifying the elements and attributes allowed in the XML document, the nesting and occurrences of each element, and any external entities. This DTD file is linked with the actual XML document by placing a tag into the XML document. The opening characters of this tag are <!DOCYPE followed by the name of the root element. Figure 4.3 shows a sample DTD.

XML Schema. The XML Schema is a newer standard seeking to improve the DTDs by adding more typing and constructs. Advantages of XML Schemas include strong data typing, modular design, namespaces support, efficient data exchange, easier to learn, and fewer errors. While DTDs can be adequate for simple XML documents, schemas provide more control over the content and structure of complex XML documents. Figure 4.4 shows a sample schema.

4.3.3 XML Parsing

XML documents are processed by special applications called parsers (Figure 4.5). The parser reads the document and generates output based on the document's content and the markup used to describe that content. The parser checks to make sure the document is well-formed. Parsers that have the ability to compare the document to its DTD or schema to determine whether the document is valid are called validating parsers. Moreover, the parser can also

```
?>xml version="1.0" encoding="UTF-8<?"
!>ELEMENT device (#PCDATA | id | measure | service<*(
!>ELEMENT id (tag, variable, manufacturer, model<(
!>ELEMENT manufacturer (#PCDATA<(
!>ELEMENT measure (unit, value, quality<(
!>ELEMENT model (#PCDATA<(
!>ELEMENT operator (#PCDATA<(
!>ELEMENT precision (#PCDATA<(
!>ELEMENT quality (#PCDATA<(
!>ELEMENT service (settingdate, operator, precision<(
!>ELEMENT settingdate (#PCDATA<(
!>ELEMENT tag (#PCDATA<(
!>ELEMENT unit (#PCDATA<(
!>ELEMENT value (#PCDATA<(
!>ELEMENT variable (#PCDATA<(
```

Figure 4.3: Sample DTD file

```
?>xml version="1.0" encoding="UTF-8<?"
>xs:schema elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema<"
      >xs:element name="device<"
            >xs:complexType mixed="true<"
                  >xs:choice minOccurs="0" maxOccurs="unbounded<"
                        >xs:element ref="identification</"
                        >xs:element ref="measure</"
                        >xs:element ref="service</"
                  />xs:choice<
            />xs:complexType<
      />xs:element<
      >xs:element name="identification<"
            >xs:complexType<
                  >xs:sequence<
                        >xs:element ref="tag</"
                        >xs:element ref="variable</"
                        >xs:element ref="manufacturer</"
                        >xs:element ref="model</"
                  />xs:sequence<
            />xs:complexType<
      />xs:element<
      >xs:element name="manufacturer" type="xs:string</"
      >xs:element name="measure<"
            >xs:complexType<
                  >xs:sequence<
                        >xs:element ref="unit</"
                        >xs:element ref="value</"
                        >xs:element ref="quality</"
                  />xs:sequence<
            />xs:complexType<
      />xs:element<
      >xs:element name="model" type="xs:string</"
      >xs:element name="operator" type="xs:string</"
      >xs:element name="precision" type="xs:decimal</"
      >xs:element name="quality" type="xs:string</"
      >xs:element name="service<"
            >xs:complexType<
                  >xs:sequence<
                        >xs:element ref="settingdate</"
                        >xs:element ref="operator</"
                        >xs:element ref="precision</"
                  />xs:sequence<
            />xs:complexType<
      />xs:element<
      >xs:element name="settingdate" type="xs:string</"
      >xs:element name="tag" type="xs:string</"
      >xs:element name="unit" type="xs:string</"
      >xs:element name="value" type="xs:decimal</"
      >xs:element name="variable" type="xs:string</"
/>xs:schema<
```

Figure 4.4: Sample XML Schema

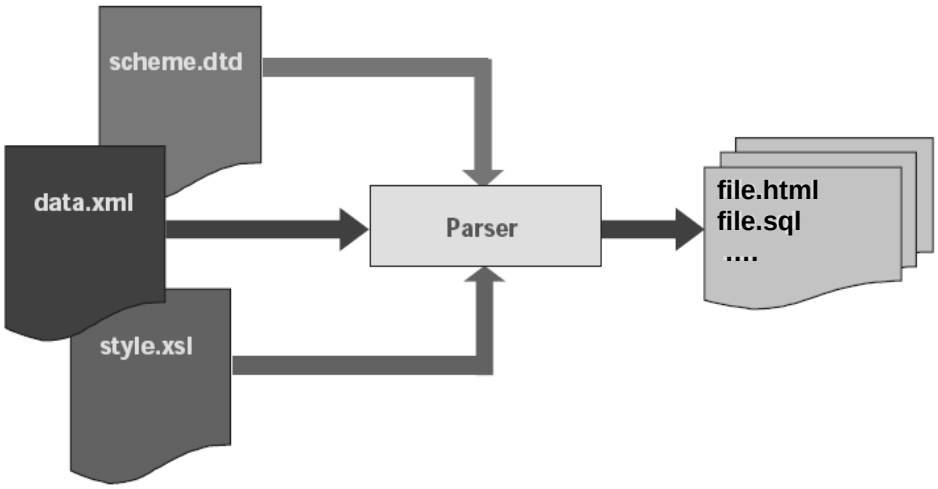


Figure 4.5: XML Parsing

process a style sheet document which defines the appearance of the XML document within a browser. Because XML is an Internet and web technology, the majority of parsers are written as Java applets.

4.3.4 XML Characteristics

Portable Data. Because XML is simply text, it can be moved between various machines regardless of their hardware platform or operating system. XML files must conform to a specification defined by the World Wide Web Consortium (W3C). Therefore, the sending and receiving applications must both adhere to the XML standard. This allows moving data between different platforms, just like Java allows moving code around different platforms.

Interoperability. In comparison to HTML, which is a tightly-defined standard aimed specifically at web presentation, XML is a generalized standard which focuses on describing the structure of the documents. The programmers have the flexibility to define the content of the document using their own elements and attributes. This allows for interoperability. Each industry can agree upon a certain set of constraints for XML, and then exchange data in those formats, allowing them to apply their business knowledge to the data being exchanged to make it meaningful.

Open Standard. XML does not require any special expertise to develop. Unlike proprietary solutions which define binary data formats that must be decoded in certain ways, and involve communication with other companies, extensive documentation, and coding efforts; XML is a proven technology that already has the tools , APIs, and parsers that handle all the work. Millions of

developers are working, fixing, and extending it every day. XML becomes a reliable, unimportant part of your application, allowing you to focus on the more important issues: the complex business logic and presentation that involves months of thought and hard work.

4.4 Java and XML Combined

4.4.1 Methods to Combine Java and XML

Low-Level APIs. The most basic way of accessing XML information from within a Java program is though a low-level application programming interface (API) [34]. Using these APIs, the textual content of the XML document can be directly accessed. This requires a strong XML knowledge from the developer's part. Since these low-level APIs deal with the document structure and XML rules more than with the actual data, they are commonly used in infrastructure tasks or when setting up communication in messaging.

Different approaches have been used to implement these low-level APIs. For example, the Simple API for XML (SAX) works with streamed data (reads information from an XML input source), the Document Object Model (DOM) works with modeled data (keeps a complete in-memory model of an XML document), and the Java API for XML Parsing (JAXP) works with abstracted data (abstracts previous types of APIs and makes them vendor-neutral.)

High-Level APIs. Instead of parsing and traversing the XML documents directly, the high-level APIs hide most of the details of XML structure and rules, and allow Java classes to work with the business logic implied within the document rather than the data. This is easier and more useful when solving

specific business problems, and has become quite popular with the developers of Java- and XML-based applications.

Different approaches have been used to implement these high-level APIs [34]. For example, the mapped data approach maps data from an XML document to Java classes, while the messaged data approach uses XML as the interchange medium for data. Examples of the latter approach include the Simple Object Access Protocol (SOAP) and XML Remote Procedure Call (XML-RPC.)

The high-level API approach is generally known as the data binding approach. This usually consists of four parts: binding schema (specifying how Java classes are generated from XML constraints), class generation (creating the Java source files from XML constraints), unmarshalling (converting XML documents to Java classes), marshaling (converting Java classes to XML documents.) This process is depicted in Figure 4.6.

4.4.2 Characteristics of Combined Java and XML

Dual Portability. Combining the code portability of Java with the data portability of XML allows the creation of powerful distributed applications.

Open Standards. Because W3C manages the XML specification and Sun Microsystems controls the Java programming language specification, and because millions of developers make changes to the standards and specifications of Java and XML only after through testing and investigation, both Java and XML have well-defined specifications. This leads to longer lifetime for any Java applications and any information stored with XML. Changes to the

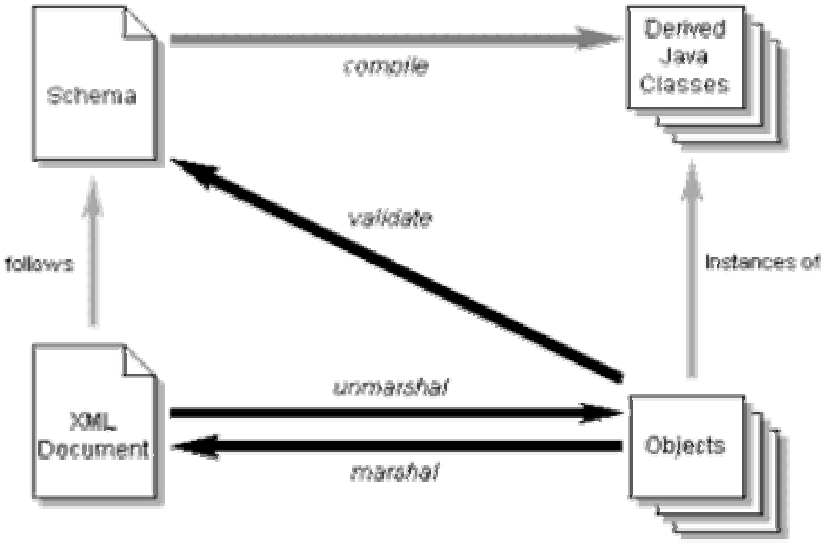


Figure 4.6: Java and XML data binding

Java and XML specifications are also more infrequent than newer technology specifications.

Extensible. Both Java and XML have built-in extensibility features. As a truly object-oriented programming language, portions of Java code can be improved, modified, or even rewritten without altering any other parts of the application. Similarly, XML information can be rearranged, sorted, or modified in whatever manner suits the applications which access the data. XML documents can utilize an unlimited number of markup tags. The document's author controls the nature of these tags. This further ensures the longest possible lifetime for the applications and the data created.

Internet-Compatible. As one of the most popular programming languages for creating network applications, Java allows building both large and small networks, especially the ones planned to be connected to the Internet. Similarly, XML information is fast becoming one of the most popular methods of storing data on the Internet, especially on the World Wide Web. Moreover, because XML is derived from SGML, the same source from where HTML was derived, many Java developers familiar with HTML can easily make the transition to Java- and XML-based information.

Interoperability. Since XML is vendor neutral, meaning that no one company controls the specification, developers of new applications and technologies are more comfortable with the concept of using XML to structure their data. XML data in itself is very easy to process. An application developed in Java can access XML documents as easily as it can access any other file. Both Java and XML use Unicode character encoding, a system that makes it easy to

exchange data and information between XML and Java code.

Reusability. Both Java applications and XML documents can be created in a modular fashion. By following a modular design, the code modules can be made available for reuse by other applications. This allows developers to quickly create flexible, reliable, and more efficient applications using Java and XML.

4.5 Applications of Java and XML in e-Business

4.5.1 Web Services

The term web service is commonly used to describe a service-oriented system architecture that is based on open and interoperable XML standards. It is defined as an application that exists in a distributed environment such as the Internet. It accepts a request, performs its action based on the request, and returns a response. Both the request and response usually take the form of XML, and are delivered over a wire protocol such as HTTP. [35, 36, 37, 38]

Web services can be applied in several different ways. They generally fall into three categories: (1) allowing programmatic access to applications over the Internet, (2) business-to-business (B2B) integration between different organizations, and (3) application-to-application (A2A) integration within the same organization.

This usually involves three main steps: (1) developing the web service and publishing its features, (2) discovering and learning about the available web services, and (3) accessing the web services and binding to it from within the client applications. This publish-discover-bind model is depicted in Figure 4.7.

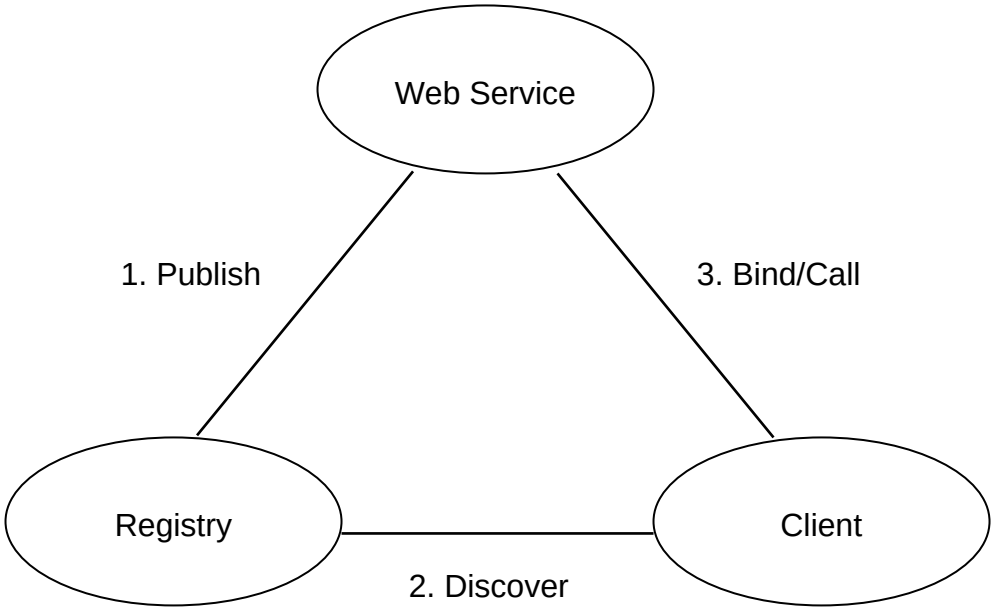


Figure 4.7: The web services' publish-discover-bind model

The core technologies that underline these three main steps are WSDL, UDDI, and SOAP/HTTP.

Web Services Description Language (WSDL). This is the mechanism that allows the provider of a web service to specify the technical details of exactly what services are offered. WSDL is used to group related operations into interfaces and then provide some way to describe each of those operations. The WSDL itself is defined using XML.

Universal Description, Discovery, and Integration (UDDI). This is the global registry of web services. It allows providers of web services to advertise their offerings in a standard way on the Internet. It also allows developers of client applications to search the available web services and learn what interfaces they provide in order to be able to build those client applications.

Simple Object Access Protocol (SOAP). Once an interface has been defined, clients must use some protocol to invoke the operations on that interface. SOAP is the most common choice. It provides a way to identify which operation to invoke, to convey that operation's inputs as XML data, and to return any output also as XML data. SOAP forms an envelope around the XML data, and then carries it over HTTP.

4.5.2 Web Services Standards

The novel idea of web services was created by a group of companies including Microsoft, IBM, Sun, Oracle, BEA, and many others. They have all endorsed the core web services technologies of SOAP, WSDL, and UDDI. Most of these technologies have been submitted to the W3C and have become official

standards of the web. The W3C standards can be referred to at http://www.w3.org.

4.5.3 Web Services Platforms

On the commercial side, two mainstream software development environments have dominated the web services platforms. The Java 2 Enterprise Edition (J2EE) is the platform provided by Sun Microsystems (Figure 4.8), while .NET is the platform provided by Microsoft (Figure 4.9).

These two competing platforms have several similarities and differences. In terms of similarities, both environments are trying to support the same classes of applications. The J2EE platform is based on the Java Virtual Machine (JVM), while the .NET platform is based on the Common Language Runtime (CLR). The large standard library that Java provides includes Java Server Pages (JSP) for web scripting, JDBC for database access, Swing for building graphical user interfaces, Enterprise Java Beans (EJBs) for building scalable server applications, and other classes. These are analogous to the .NET platform's ASP.NET, ADO.NET, Windows Forms, and Enterprise Services, respectively.

In terms of differences, the Java environment uses one programming language (Java) which can run on diverse operating systems, while the .NET platform provides a variety of programming languages (Visual Basic.NET, C++, C#) but focuses on the Windows operating system. The Java environment provides portability (which is good) but with less integration with the operating system, while .NET is not portable but is tightly integrated to Windows (which is good also).

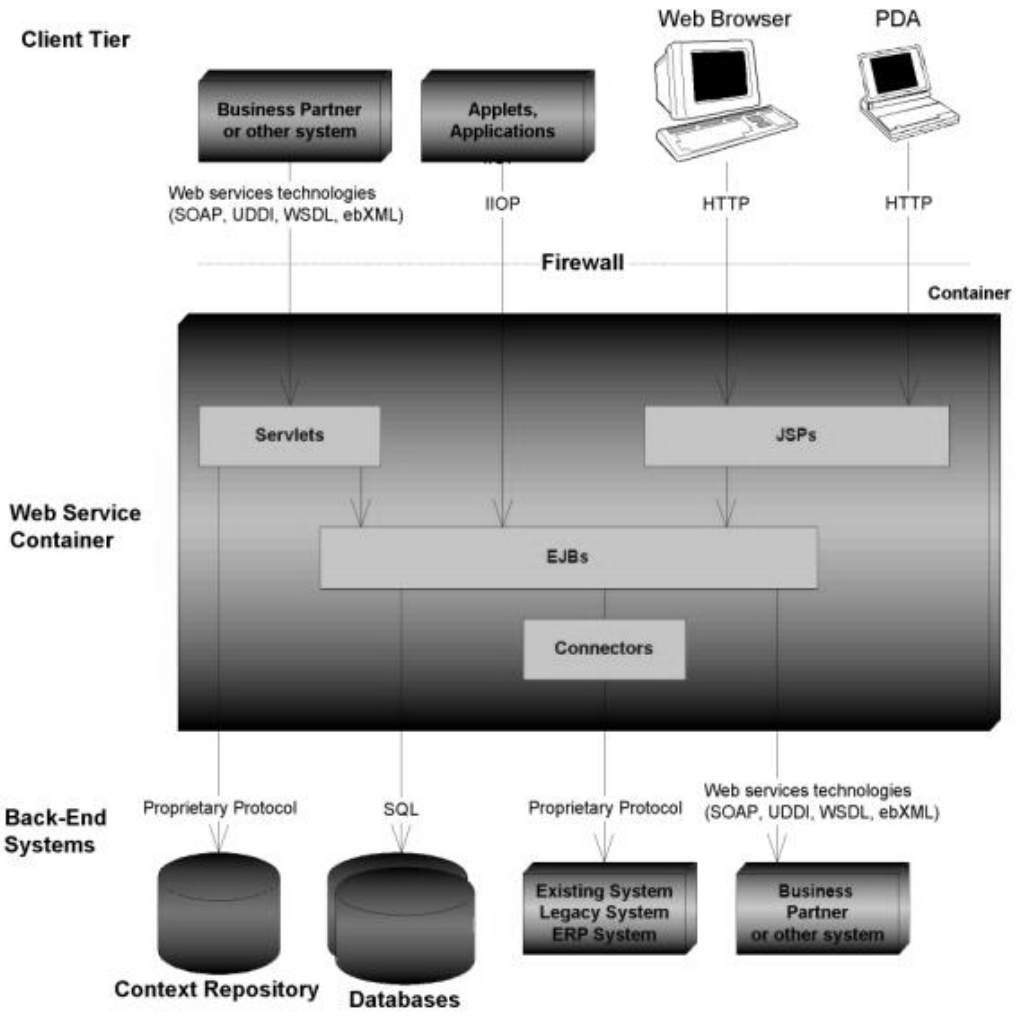


Figure 4.8: Sun J2EE platform

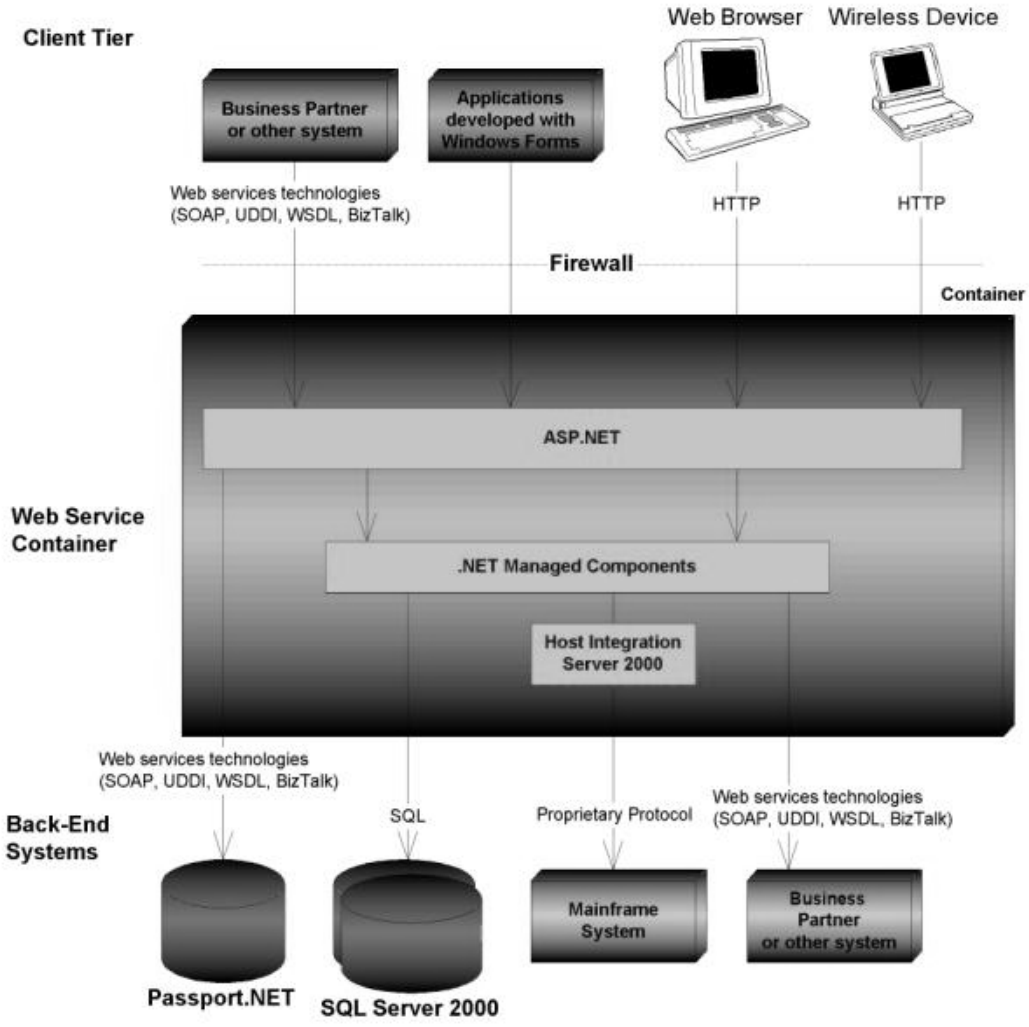


Figure 4.9: Microsoft .NET platform

4.6 Applications of Java and XML in Process Automation

The effect of XML and web services technologies on the process automation field has been shown in some applications [39, 40, 41]. They can generally be summarized into the following:

4.6.1 Web Enabling

The term web enabling is used here to refer to the process of applying a web interface to a control process. Java and HTML have been the main technologies to develop web enabled process automation systems. In Chapter 5, we demonstrate these concepts by describing the design of a web enabled control system that was developed as part of a lab-scale control project.

4.6.2 Web Integration

The term web integration is used here to refer to the process of integrating between different distributed subsystems using standard web technologies. Java and XML can play a major role in this area. In chapter 6, we demonstrate these concepts by describing the design of a web integrated SCADA system.

CHAPTER 5

DESIGN OF WEB-ENABLED CONTROL SYSTEM

In this chapter, we provide an overview of a web-enabling project which was conducted by the Systems Engineering Department at KFUPM [1]. The objective of this project was to design and develop a Java-based control system that will enable the use of laboratory equipments from remote locations. Figure 5.1 summarizes the project main goal.

The system architecture consists of a server computer, called the LabStation, and several remote client computers, called the RemoteUser PCs. The LabStation server is connected to the lab equipment using data acquisition card, digital signal processing board, and other add-on controller hardware. The Java Native Interface (JNI) technology is used for accessing the I/O ports, and Java servlets are used to provide functions required by the remote user. The RemoteUser PC provides access to the LabStation server through Java applets running through the standard web browser interface.

5.1 Dual Tank Process

The process under control is the CE105 dual tank system provided by TecQuipment, shown in Figure 5.2. This apparatus is used for the study and practical investigation of basic and advanced control engineering principles,

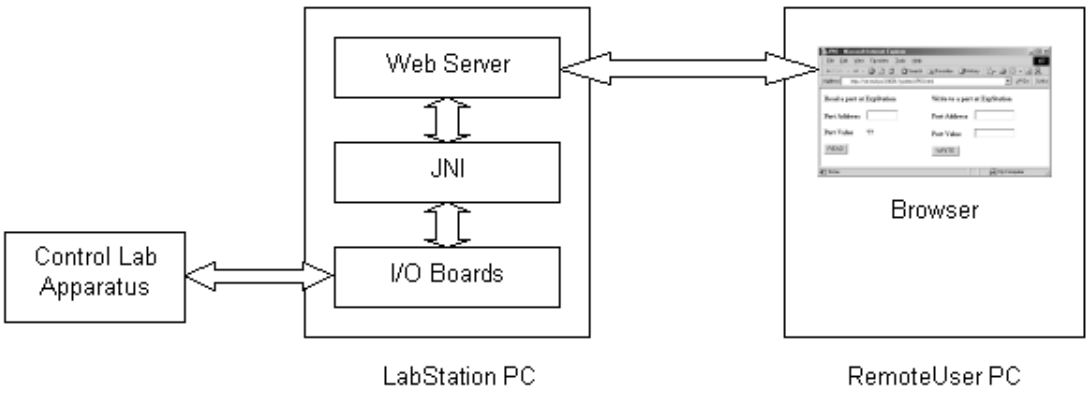


Figure 5.1: A lab-scale Java-based control system

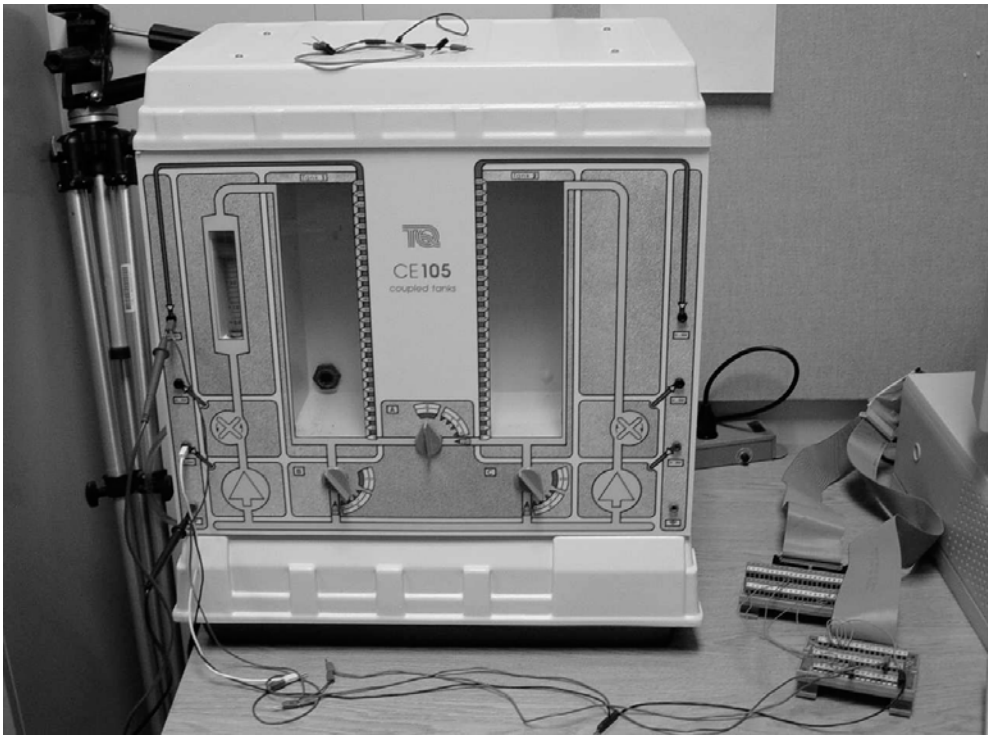


Figure 5.2: The CE105 dual tank system

including static and dynamic systems behavior using either analog or digital techniques. The CE105 coupled tanks show the fluid transport and liquid level control problems as they would occur in process control industries. It may also be used for designing and implementing three-term PID controllers.

The basic control problem is to regulate the liquid level in one of the tanks by varying the speed of the circulating pump. The CE105 comprises two separate tanks, interconnected by a flow channel, both fitted with drain valves to a common reservoir situated below. A variable area valve in this channel is used to vary the flow characteristics between the tanks. Pressure transducers provide 0-10V output signals to indicate actual water level in each tank. A variable speed pump is set to fill the left-hand tank, either under manual or automatic control, and the second tank can be filled from the first tank, via the variable area valve. The pump flow rate is indicated by a flow meter and impulse transducer. The water level in each tank is visible through the front panel windows which are marked with calibration scale.

5.2 System Components

The LabStation server is a Windows-based personal computer. Since the control application was developed in Java, the operating system could have been replaced with Unix or any other platform as long as it supports Java. The main add-on components to this server are explained below.

5.2.1 Data Acquisition (DAQ) Card

The PC-30F/G board from Eagle Technologies was used to provide analog

and digital I/O interface with the CE105 dual tank system. This is an ISA-bus multi-function board, programmable gain, simultaneous sample and hold, 4 analog outputs, 16 single ended analog inputs or 8 differential analog inputs, and flexible digital I/O capabilities (24 lines in three ports, each port can be configured as input or output.) The board also includes a 16-bit counter/timer used to generate or measure frequency and count events.

5.2.2 Digital Signal Processing (DSP) Card

The PC32 board from Innovative Integration was used to provide digital signal processing capabilities on a single half-length ISA bus card. The PC32 offers a very high-performance computing engine with determinant timing for fast data acquisition and control to complement the personal computer. Using the PC32, the server is freed up from time-critical events and can operate under Windows and other operating systems where real-time performance is not very easy to achieve.

5.2.3 Video Capture Card

In this project, an industrial digital camera was used to provide visual feedback to the remote user. The video capture board used is from Prolink and is based on Brooktree/Conexant Bt878 video decoder chip. The Bt878 chip has two high speed A/D converters capable of doing 40 X 10^6 A/D conversions per second.

5.3 Java Development

5.3.1 Java Drivers

Although this project used special hardware components such as the data acquisition card and digital signal processing card, achieving hardware independence was very important. To do this, an abstraction layer was needed to establish communication between the I/O cards and the control application. This was possible using the Java Native Interface (JNI). However, it was assumed that a minimum set of features will always be available on any DAQ and DSP cards. This usually can be found as C libraries on the card which can be accessed by the JNI layer.

The minimum functions required for the DAQ card include initialize/ reset, get total number of inputs/outputs, get minimum and maximum of input/output channels, read input channels, write to output channels, and streaming data collection. The minimum functions for the DSP card include initialize/reset, upload a compiled program, read input channels, and write to output channels. The Java drivers development includes both generating Java files with native method declarations, and C/C++ files with native method implementations. The complete implementation details are included in the project report [1] and are not part of this thesis work.

5.3.2 Java Servlets

The interaction between the remote users and the LabStation server is handled through the Java servlet technology (Figure 5.3). The servlets respond

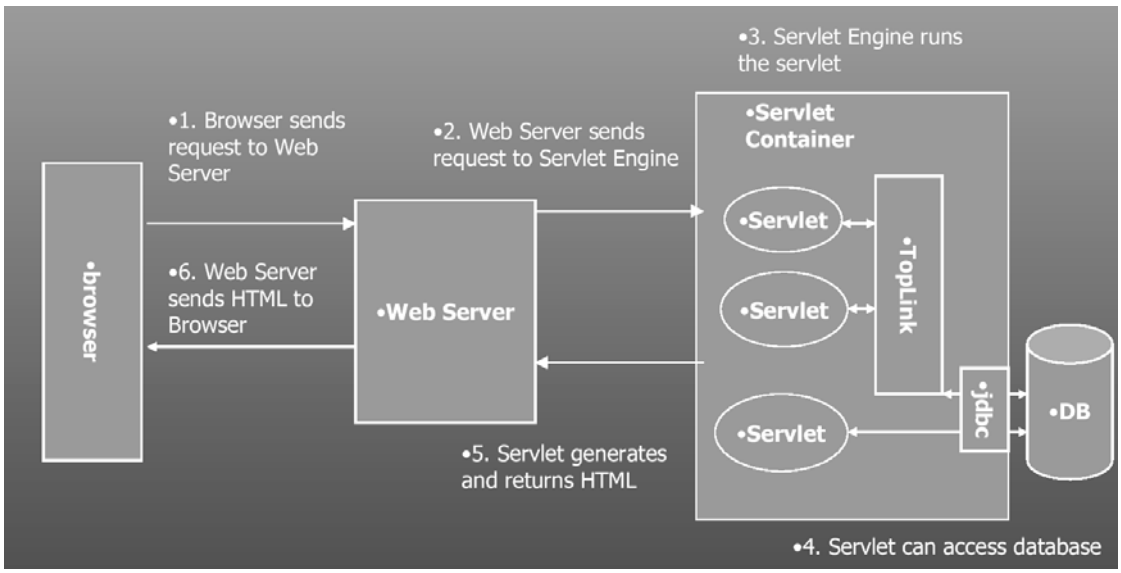


Figure 5.3: The servlet processing mechanism

to requests sent from applets running on the RemoteUser PC, process the required actions to the I/O drivers (initialize/reset, input, output, upload compiled programs, etc.) and respond accordingly through the web server to the remote user applets.

The LabStation servlets and classes that were developed for this project include an experiment control servlet, a remote procedure call servlet, a DSP program uploader servlet, a system event logger class and a resource access serialization class.

5.3.3 Java Applets

The main graphical user interface (GUI) applet for this project is called TeleLabJApplet (Figure 5.4). It is based on the JFC (Java Foundation Classes)/Swing components which include a rich set of features such as pluggable look and feel, robust event-handling, graphics and imaging tools, window layout managers, and data transfer classes. The TeleLabJApplet includes buttons that will invoke LabStation servlets to start an experiment, change controller values, login/logout, etc. The invoked servlet returns some HTML output, or certain I/O errors due to network problems.

Moreover, additional Java classes and beans were developed for the LiveCam video portion of the user interface. These include an image painting bean, a thread manager bean, a visual interface bean, a class for copying downloaded images into memory, and a class for showing the image in a dedicated frame.

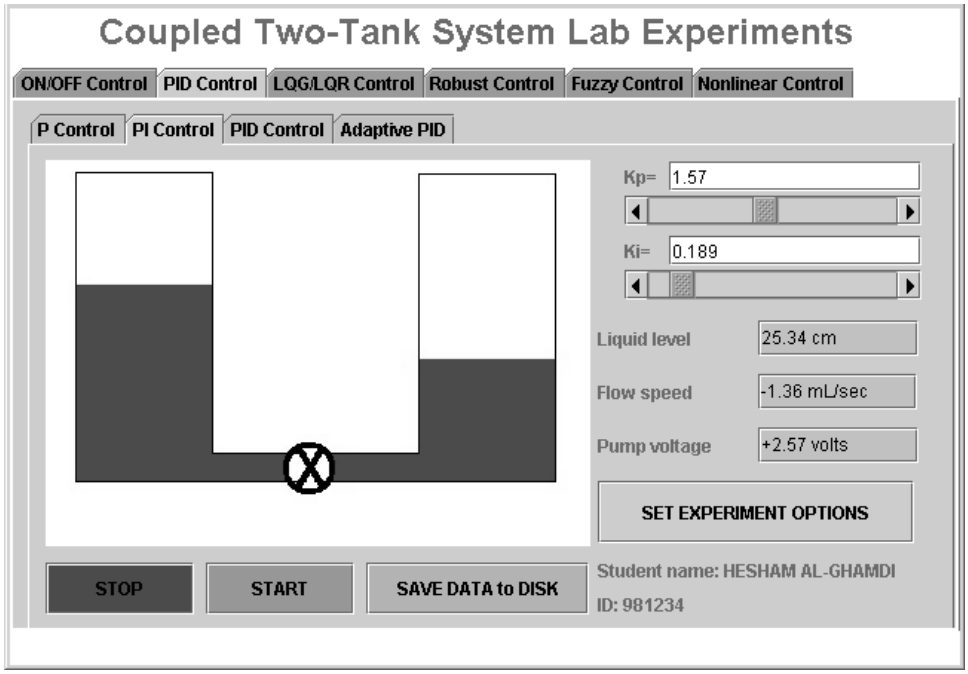


Figure 5.4: The graphical user interface applet

CHAPTER 6

DESIGN OF WEB INTEGRATED SCADA SYSTEM

In this chapter, we will discuss the design of the project conducted as part of this thesis work. While the first project discussed in Chapter 5 showed the use of Java in controlling a single lab apparatus, this second project extends and builds upon these concepts by demonstrating how Java and XML can be used together to control a distributed SCADA application.

The objective of this project is to control a hypothetical industrial process, namely the liquid transfer between two tanks. This situation can be found in many industrial applications, such as transferring products from tank to tank within a single plant, or from tank to tank between different plants across a pipeline. Figure 6.1 shows the project main goal.

Since this project is aimed at designing the high-level system architecture, the unified modeling language (UML) notation [42, 43, 44] was used to describe the design model. The UML approach provides a standard method for software development and allows the user to visualize the system design in simple and easy to understand graphical diagrams.

6.1 Tank Transfer Process

The process under control is a simple pipeline tank transfer system with

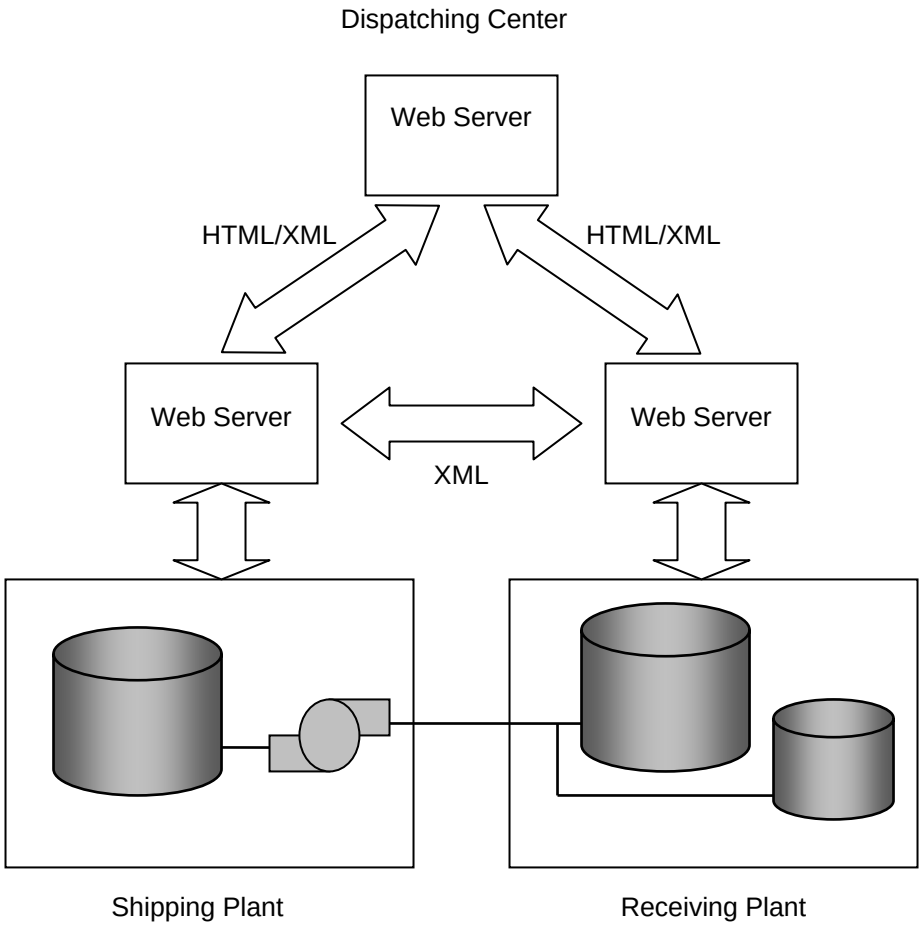


Figure 6.1: Overview of the tank transfer system

shipping and receiving facilities at both ends of the pipeline. The process consists of a shipping plant, a receiving plant, and a dispatching center. The shipping plant has the product tank and a shipping pump. The receiving plant has a receiving tank and a surge tank.

The dispatching center supervises the whole shipment process, and is responsible mainly for the business order handling. The shipping and receiving plants perform the actual field operations, and are responsible for both business and control functions. The following sections describe the UML diagrams that were developed to document the design of the overall SCADA system.

Basically, each of the three locations will be implemented as a web server. The dispatching node can be thought of as the master terminal unit (MTU) which is supervising the whole shipping operation, while the shipping and receiving nodes can be thought of as the remote terminal units (RTUs) which are interfaced to the field instrumentation.

6.2 Use Case Diagram

The use case diagram (Figure 6.2) shows the various functions required from the system. The dispatcher at the dispatching center can create a shipment order, monitor the shipment process, and close the order when it is completed. The operator at the shipping plant will receive the shipment order, initiate the shipping sequence, monitor the shipment process, and stop the shipping process once the ordered volume is completed. The operator at the receiving plant can prepare the facility to receive the shipment, monitor the shipment process, and stop the receiving activity one the ordered volume is completed.

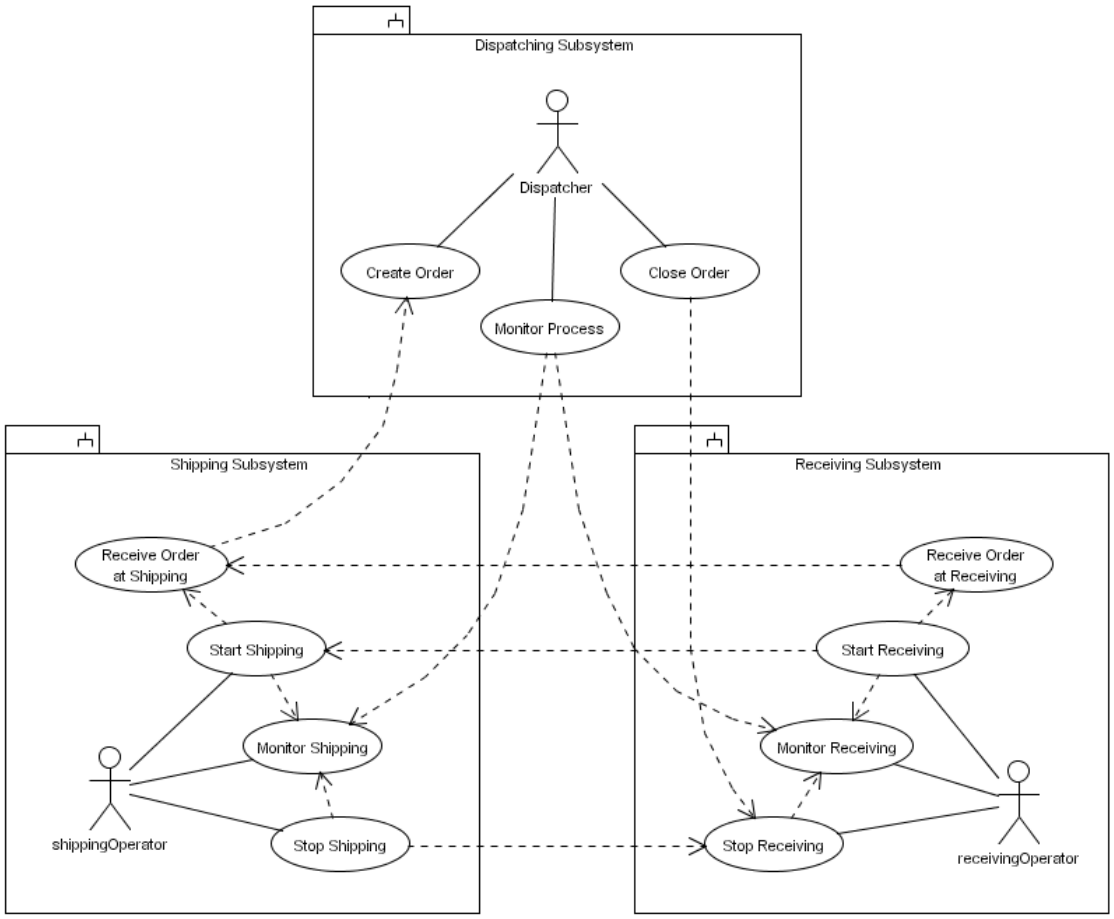


Figure 6.2: Use Case Diagram

6.3 Component/Deployment Diagram

The component/deployment diagram (Figure 6.3) shows the hardware and software components of the system. The diagram also shows what messages are sent between the different nodes on the network. Since the objective of this design project is to demonstrate the key concepts of applying Java and XML in SCADA systems and not to do actual implementation, the system components have been left as generic as possible. And since Java and XML are both portable, the developer could choose whatever hardware platform is desired.

Each of the three nodes has both business and control functionalities. These are designed as business and control web services, which will allow easy access via XML.

6.3.1 Document-Oriented Web Services

The business web services are designed as document-oriented web services, i.e. they exchange data in the form of XML documents sent over regular HTTP. An HTTP servlet handles incoming documents and passes them to the business web service.

6.3.2 RPC-Oriented Web Services

The control web services are designed as RPC-oriented web services, i.e. they invoke methods on each other remotely using SOAP/HTTP messages. An interface to the control web service handles the incoming SOAP messages for remote method invocation, while local proxies of the other remote web services are used to send outgoing SOAP messages.

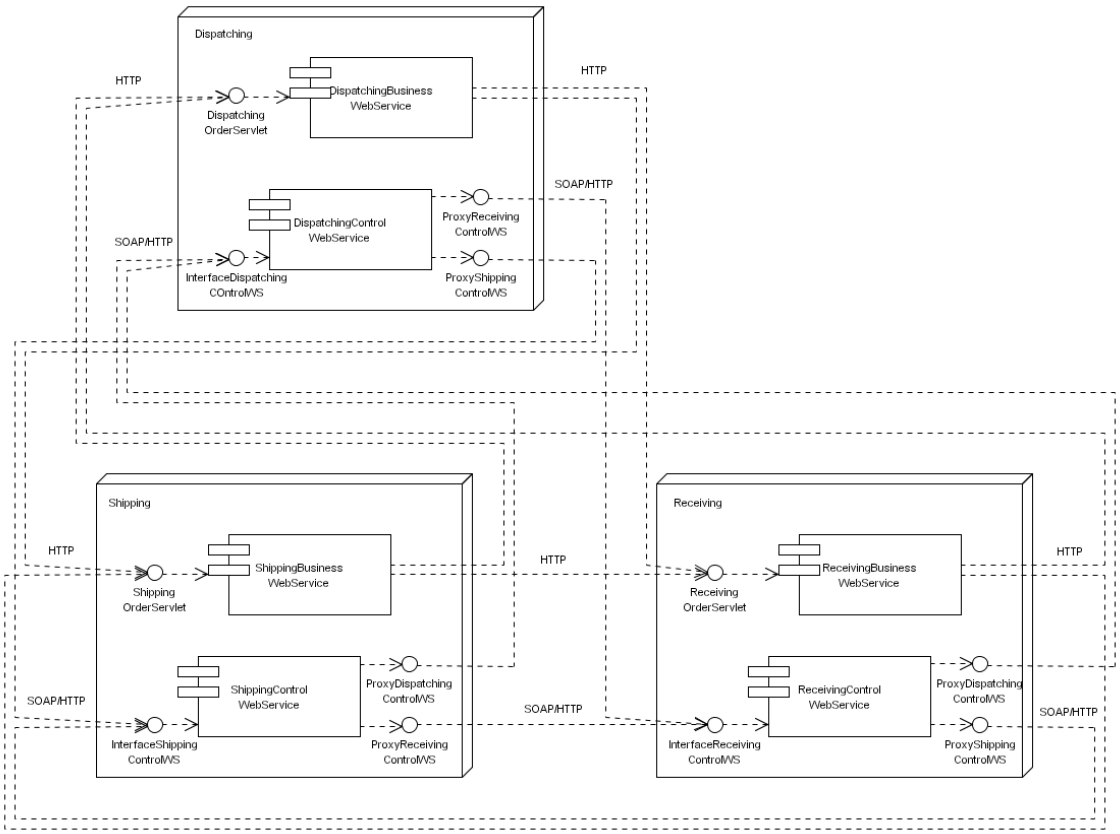


Figure 6.3: Component/Deployment Diagram

6.4 Class Diagram

The class diagram (Figure 6.4) shows the static model of the system. It shows what object classes are used and their relation to each other. This consists of the process entities, the Java drivers, the Java servlets, and the Java Applets.

6.4.1 Process Entities

There are three classes that represent physical process objects; namely the Tank, Pump, and Order classes.

Tank Class. The tank object class represents a simplified model of a tank. Each tank has an inlet valve and an outlet valve, which can be opened or closed. It also has level and volume indications, which can be checked by the control program at any time. The control program keeps scanning the process inputs and outputs (I/O) to monitor the status of the tank.

A separate class, called LevelToVolume, has been created to represent the calculation module for converting tank level to volume. This is because the tank field instrumentation often provides level sensing capability only. To get the volume of liquid inside a tank, a strapping table is used to convert the height of the liquid to a corresponding volume. This strapping table is established at the time of tank construction and is based on actual measurements of the tank.

Pump Class. The pump object class represents a simplified model of a pump. The pump can be started or stopped. The status of the pump, running or stopped, can be checked by the control program at any time. The control program keeps scanning the process inputs and outputs (I/O) to monitor the status of the pump.

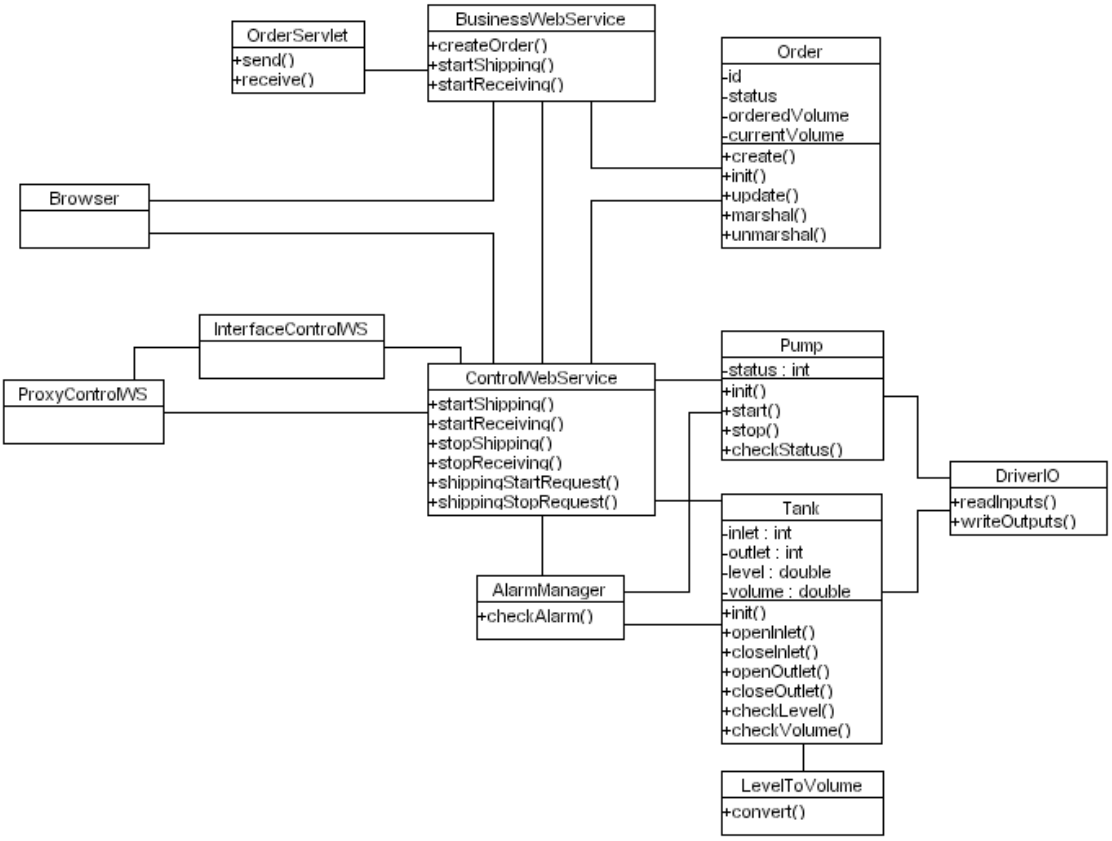


Figure 6.4: Class Diagram

Order Class. The order object class represents a model of the business order. For each transfer shipment, an order is created by the dispatcher. This order consists of an order id number, the ordered volume that needs to be transferred, the current volume shipped so far, and the order status indication. The order is downloaded from the dispatching system to each of the shipping and receiving plants, and the order status is updated as the shipment process progresses.

6.4.2 Java Drivers

In order for the tank and pump objects to interact with the physical process equipment and field instrumentation, a driver class must be created to handle the I/O interface. In this project, we created a single generic object class, called DrvierIO, to represent the I/O interface. This could represent the DAQ or DSP cards that were used in the first project, or it could represent a more sophisticated I/O modules typically found with SCADA and PLC systems. The DrvierIO simply allows the Java application to read inputs or write outputs to the field.

6.4.3 Java Servlets

The Java applications that need to interact with the other remote applications are modeled as web services, namely the BusinessWebService and ControlWebService; while the Java applications that run locally on the machine and do not require interaction with external applications are modeled as normal Java applications, namely the AlarmManager.

BusinessWebService. This web service allows the user to create an order, initiate the shipping process, and stop the shipping process. This is a document-oriented web service, which exchanges the order data in the form of XML document. It works together with the OrderServlet to send and receive the XML documents, and handles the unmarshalling and marshaling to and from a Java object.

ControlWebService. This web service allows the user to start and stop shipping, start and stop receiving, and interact with the tank and pump objects. This is an RPC-oriented web service, which allows remote invocation of methods. It has both a proxy and interface. The ProxyControlWS is used by the application sending the SOAP messages, allowing it to invoke methods on the remote web service as if it is done locally. The InterfaceControlWS is used by the application receiving the SOAP messages, allowing it to process the incoming requests and respond to them accordingly.

AlarmManager. This is a generic object class that is used to represent a subsystem handling any alarm conditions that occur from the process. The AlarmManager and the ControlWebService both monitor the process entities continuously to detect any change in value or condition.

6.4.4 Java Applets

The human-machine interface (HMI) in this project is modeled as a single generic object class, called Browser. This allows the user to interact with the SCADA system through the standard web browser.

6.5 Sequence Diagrams

The sequence diagrams show the dynamic model of the system. They show how the runtime objects act out a use case by sending messages to each other. They are used in this project to model two main processes: the monitoring process (Figures 6.5 – 6.8) and the shipping process (Figures 6.9 – 6.23.)

6.5.1 Monitoring Process

This process starts as soon as the user starts up the SCADA system. Figures 6.5 shows the overall monitoring process, while Figures 6.6, 6.7, and 6.8 show the individual sequence diagrams for the dispatching, shipping, and receiving nodes, respectively.

On the shipping and receiving nodes, the ControlWebService starts by creating the runtime objects of tanks and pump. Once the objects are created, they are updated continuously by scanning the I/O's. Both the ControlWebService and AlarmManager keep monitoring the status of the objects, and then update the Browser object accordingly.

On the dispatching node, the dispatchingControlWebService interacts with the remote shippingControlWebService and receivingControlWebService to update and synchronize the information.

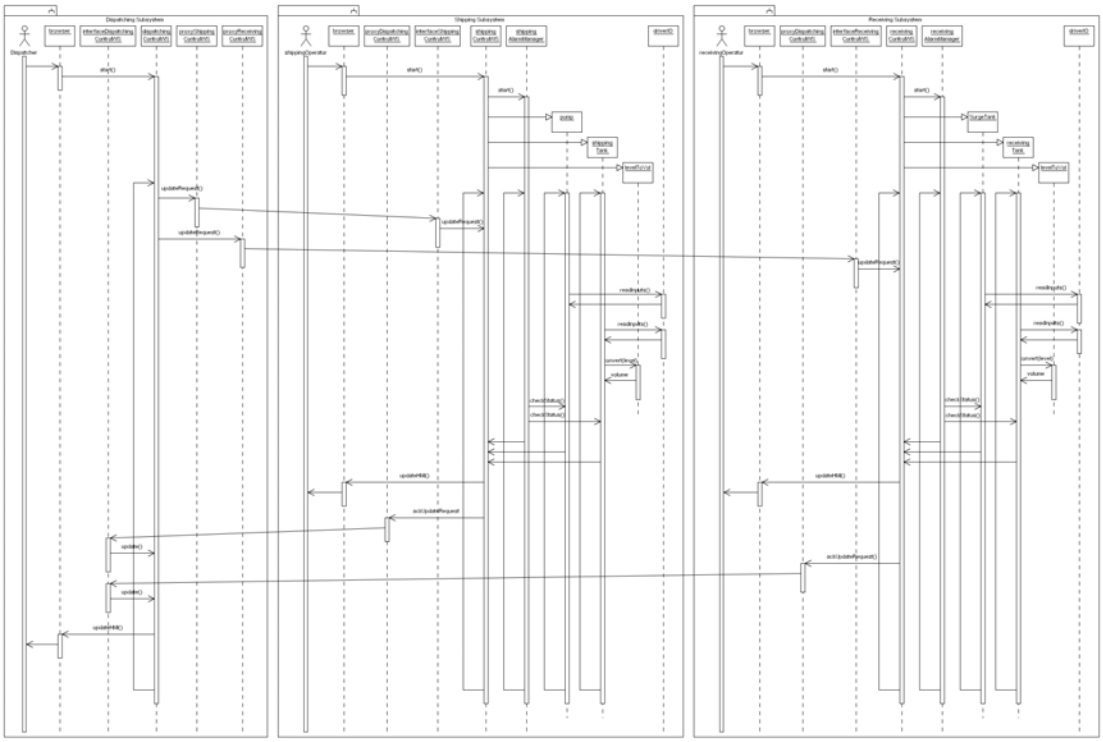


Figure 6.5: Sequence Diagram (Overall View of Monitoring Process)

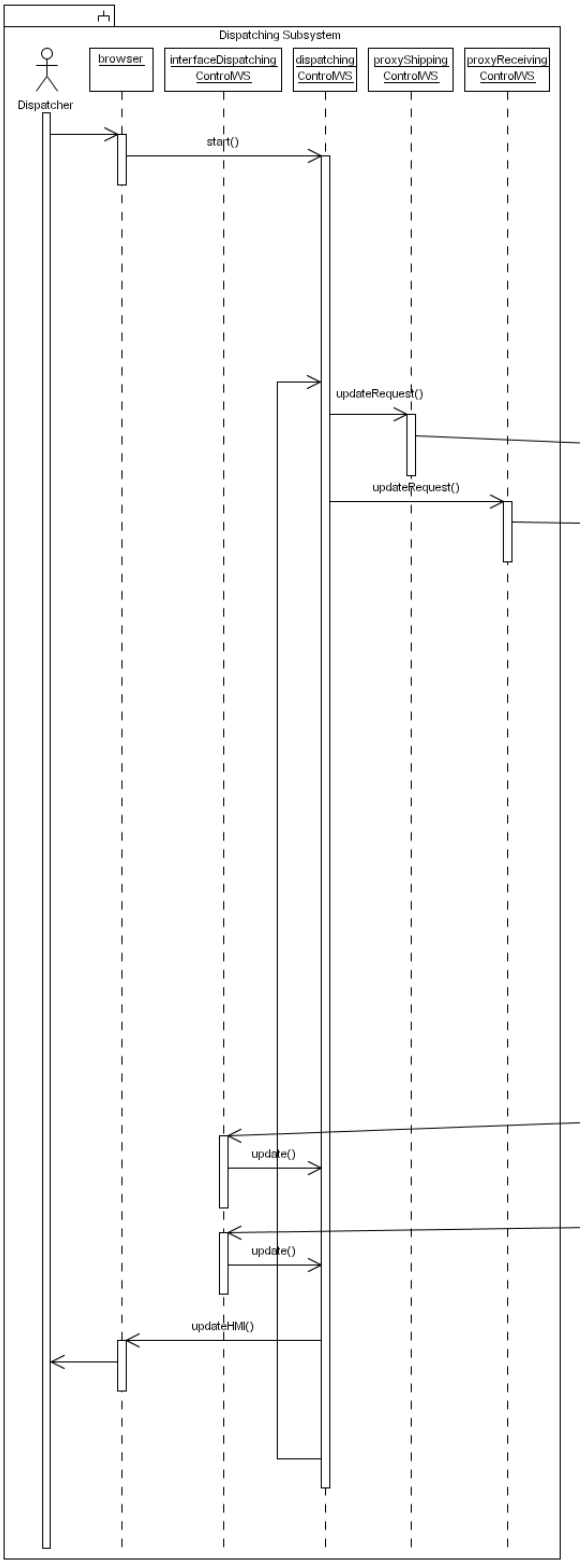


Figure 6.6: Sequence Diagram (Monitoring Process on Dispatching Node)

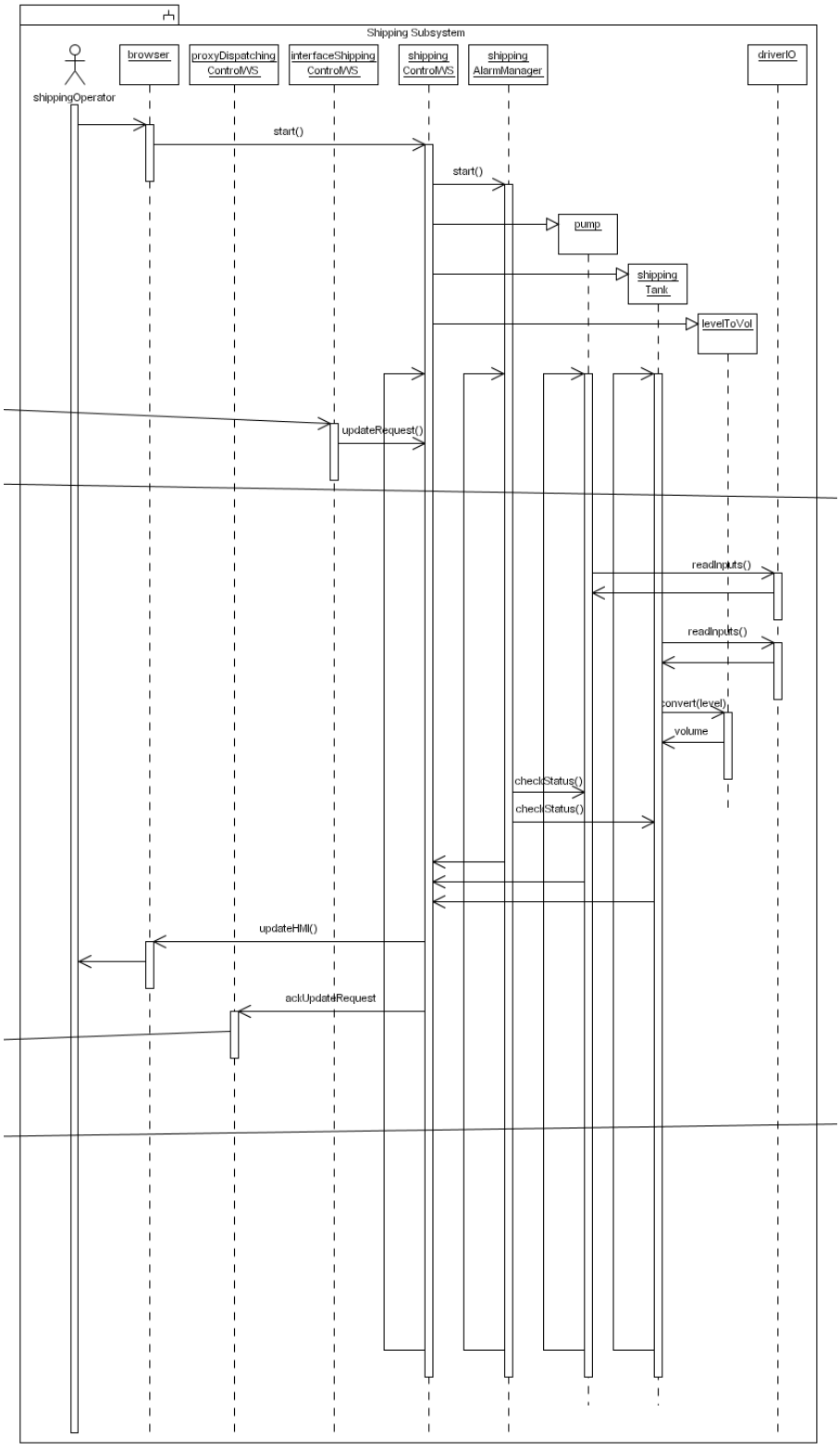


Figure 6.7: Sequence Diagram (Monitoring Process on Shipping Node)

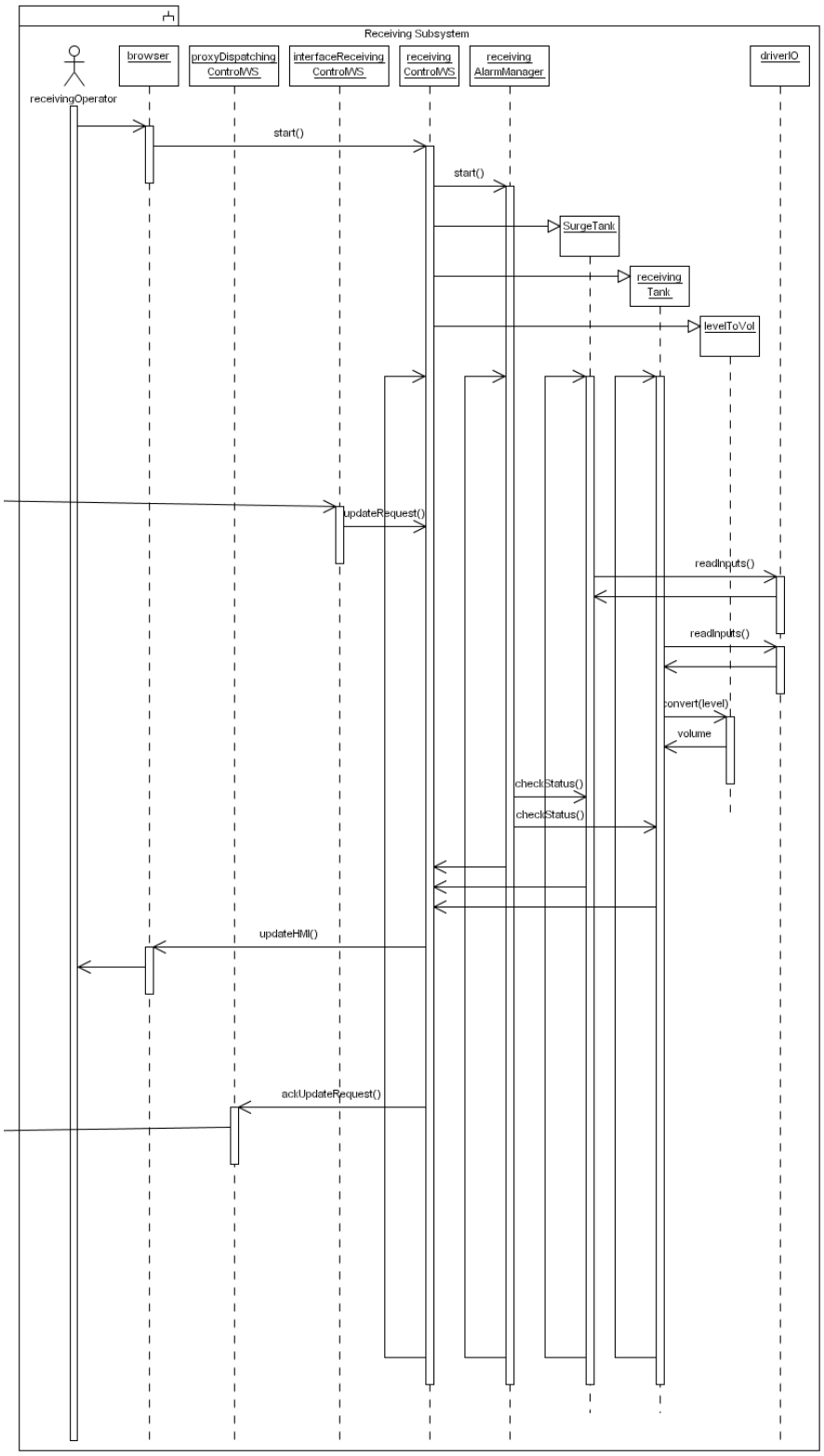


Figure 6.8: Sequence Diagram (Monitoring Process on Receiving Node)

6.5.2 Shipping Process

This process is divided into four sub-processes: create order, start shipping, stop shipping, and close order. Figure 6.9 shows the overall shipping process. Figures 6.10 – 6.13 show the order creation process. Figures 6.14 – 6.16 show the start shipping process. Figures 6.17 – 6.19 show the stop shipping process. Figure 6.20 – 6.23 show the order closure process.

The first three processes (create order, start shipping, stop shipping) require some sort of human intervention. This is just an assumption in this project to ensure the safety of critical operations such as starting and stopping a pump. The last process (close order) occurs automatically once the ordered volume is delivered completely to the receiving plant.

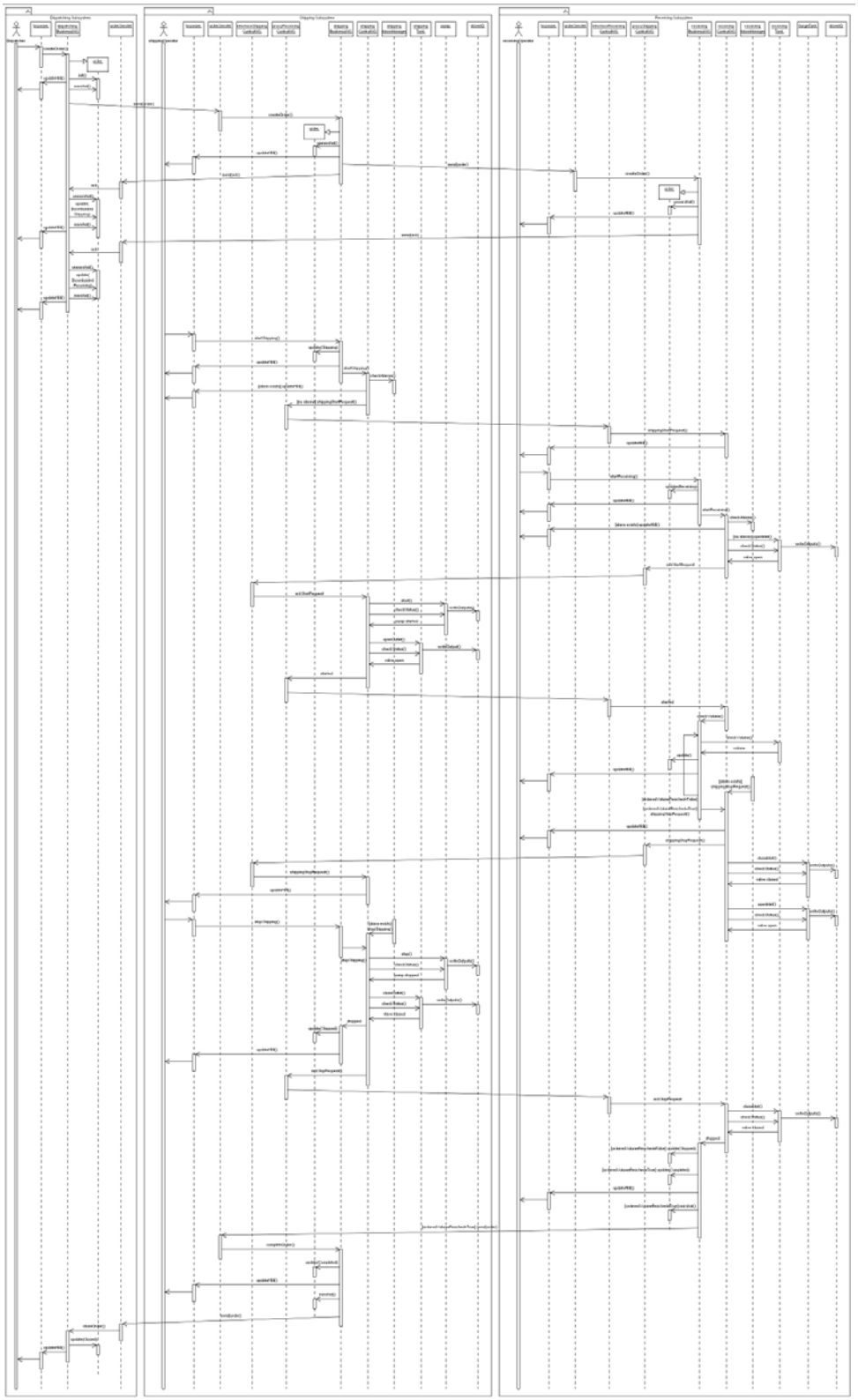


Figure 6.9: Sequence Diagram (Overall View of Shipping Process)

Create Order Process. To summarize the sequence of order creation process, the following steps are implemented:

1. The dispatcher creates a new shipment order from the browser.
2. The dispatchingBusinessWebService creates the order object.
3. The dispatchingBusinessWebService marshals the order object into an XML document and downloads it to the shipping plant.
4. The shippingBusinessWebService receives the XML document through the orderServlet and unmarshals it into an order object.
5. The shippingBusinessWebService sends a copy of the XML document to the receiving plant.
6. The receivingBusinessWebService receives the XML document through the orderServlet and unmarshals it into an order object.
7. Receiving plant sends back acknowledgment to shipping plant.
8. Shipping plant sends back acknowledgment to dispatching center.
9. The dispatchingBusinessWebService updates the order status.

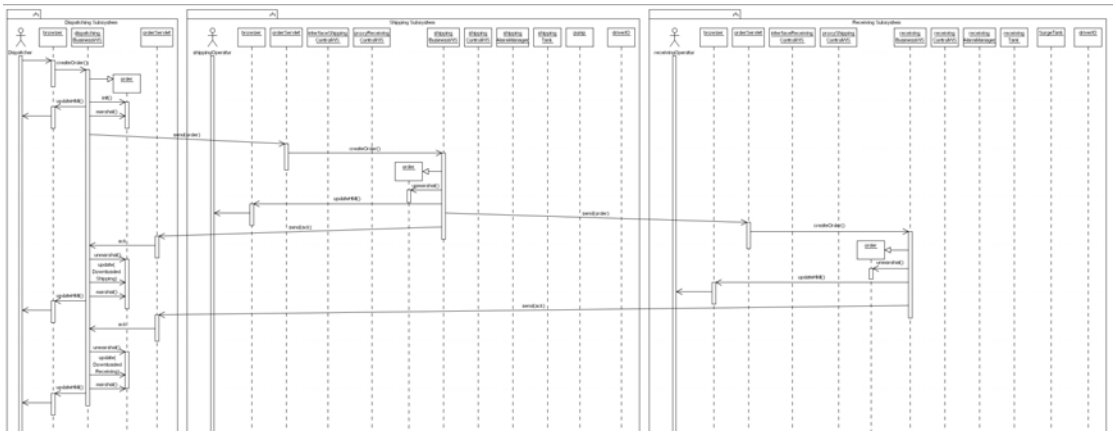


Figure 6.10: Sequence Diagram (Overall View of Create Order Process)

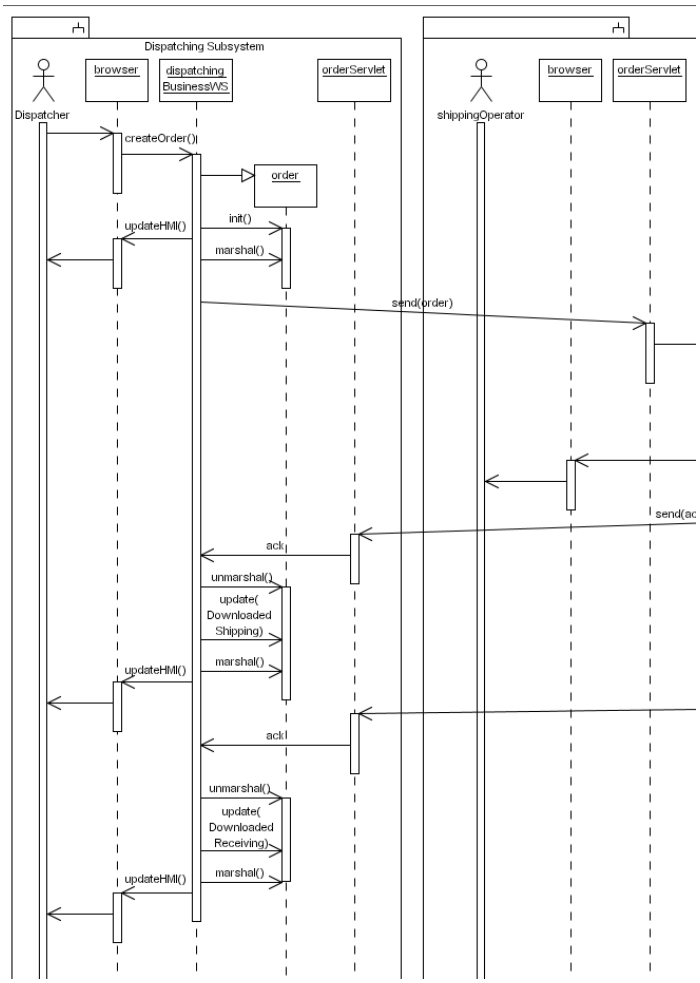


Figure 6.11: Sequence Diagram (Create Order)

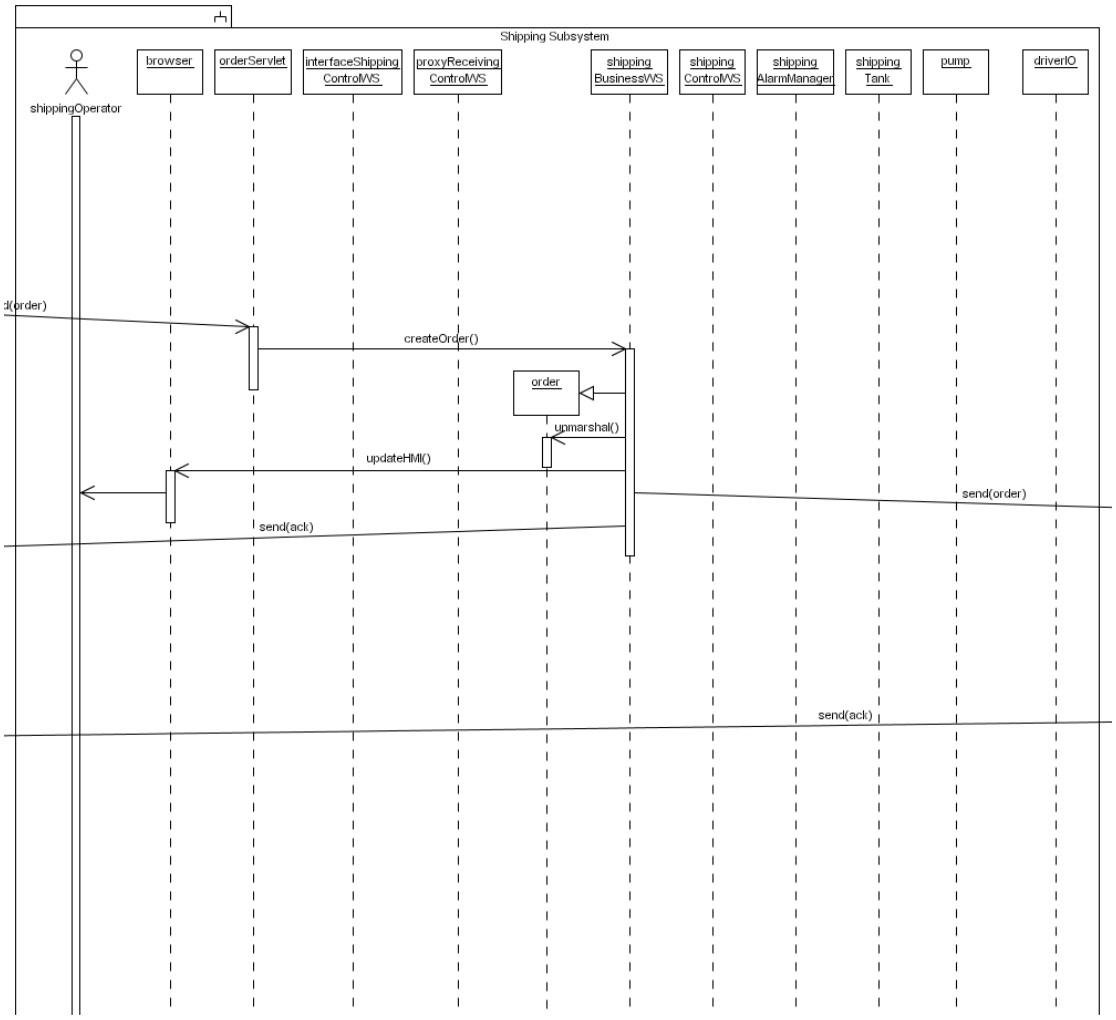


Figure 6.12: Sequence Diagram (Create Order – Continued)

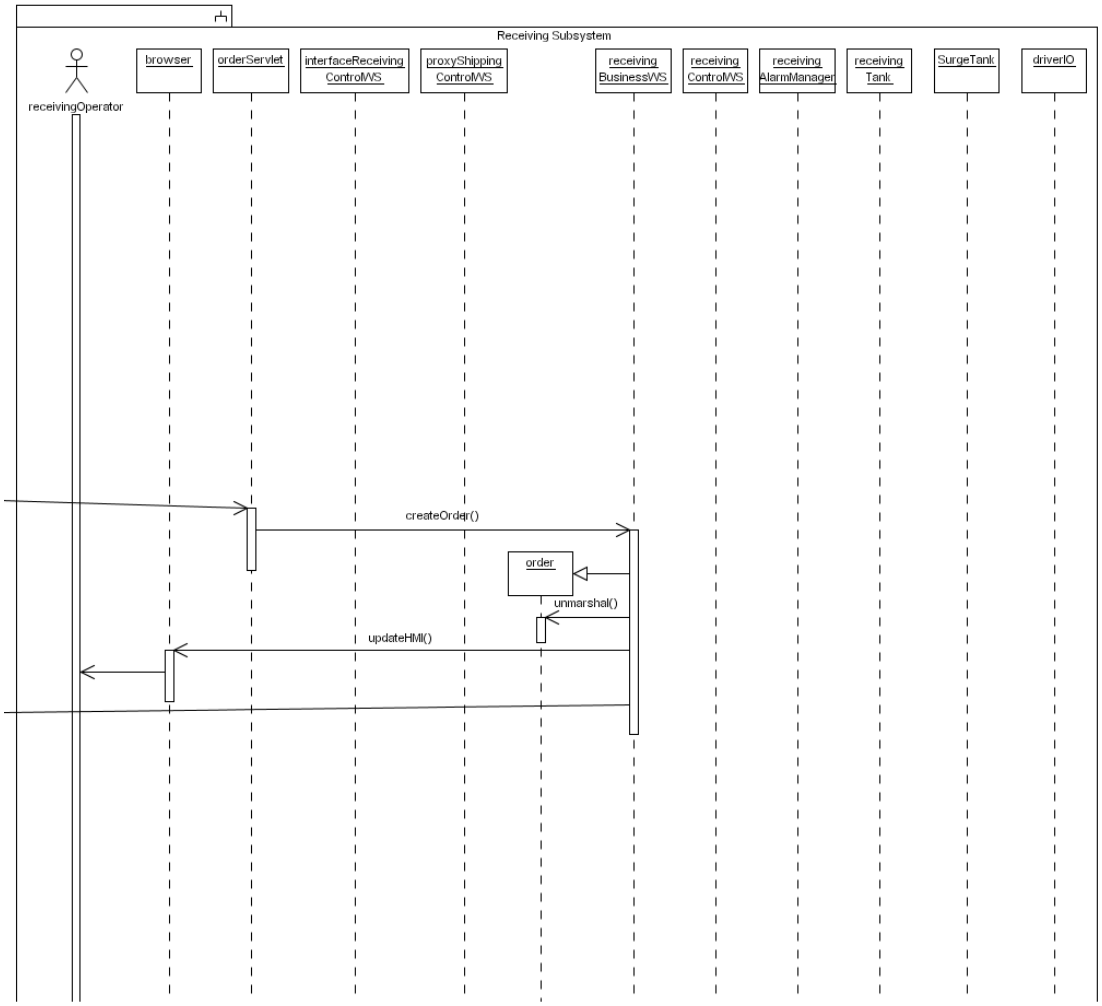


Figure 6.13: Sequence Diagram (Create Order – Continued)

Start Shipping Process. To summarize the sequence of start shipping process, the following steps are implemented:

10. The shipping operator initiates the shipping process from the HMI browser. This causes the shippingControlWebService to automatically send a start shipping request to the receiving plant.
11. The receiving operator sees the start shipping request on the HMI browser and accordingly opens the receiving tank inlet valve. This causes the receivingControlWebService to automatically send back an acknowledgement message to the shipping plant.
12. The shippingControlWebService opens the shipping tank outlet valve and starts the pump.

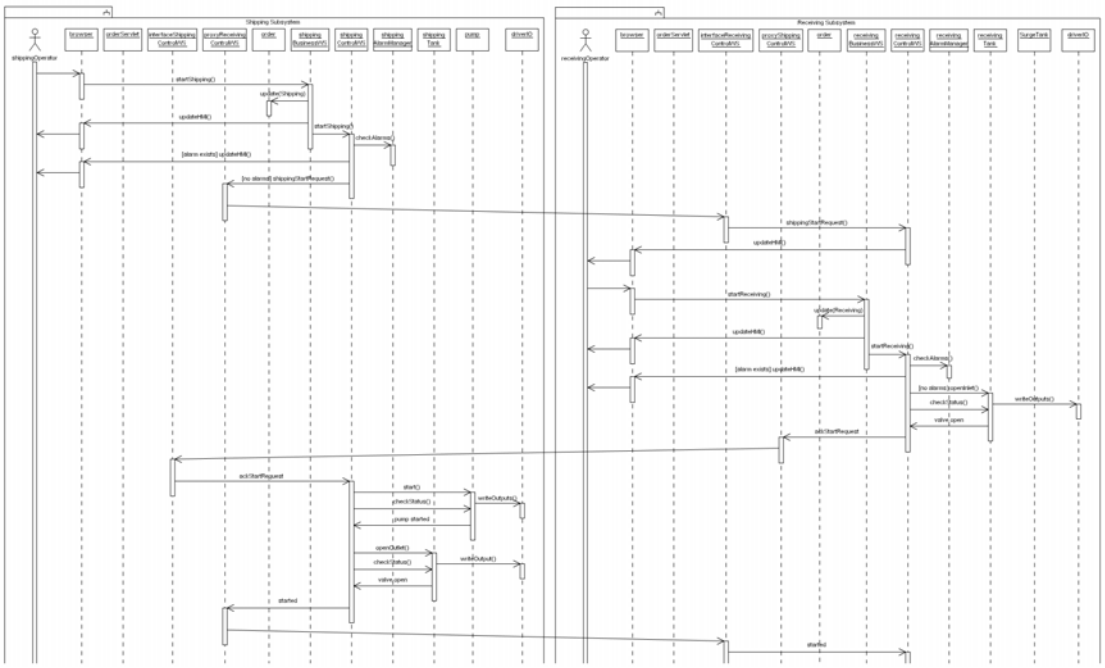


Figure 6.14: Sequence Diagram (Overall View of Start Shipping Process)

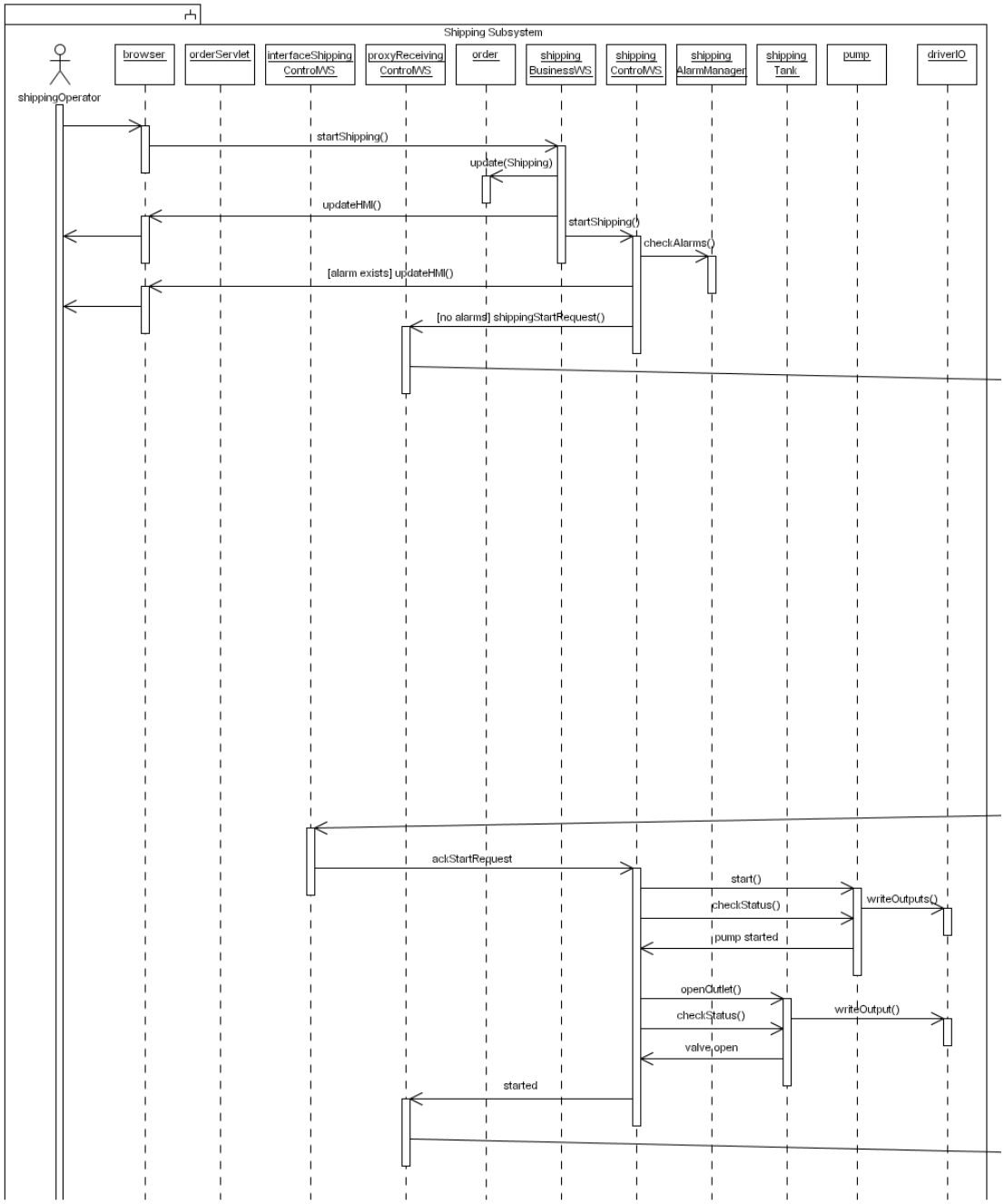


Figure 6.15: Sequence Diagram (Start Shipping)

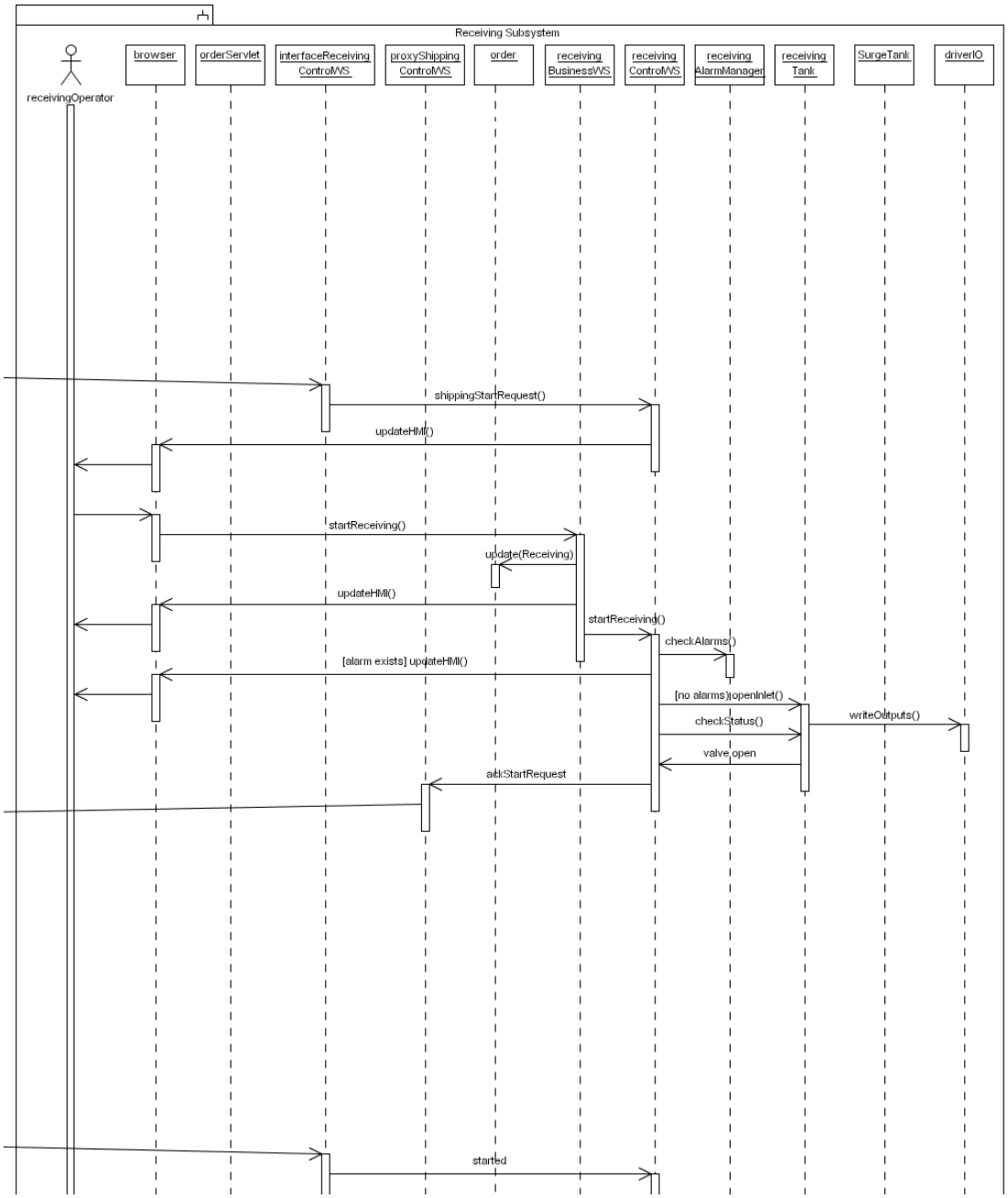


Figure 6.16: Sequence Diagram (Start Shipping – Continued)

Stop Shipping Process. To summarize the sequence of stop shipping process, the following steps are implemented:

13. When the ordered volume approaches its completion, the receiving ControlWebService sends a stop shipping request to the shipping plant.
14. The shipping operator sees the stop shipping request on the HMI browser and accordingly closes the shipping tank outlet valve and stops the pump. This causes the shippingControlWebService to automatically send an acknowledgment to the receiving plant.
15. If the ordered volume is reached and the shipping operation is not stopped yet, the receivingControlWebService switches the receiving to the surge tank so that the volume in the receiving tank is not overfilled.
16. If the shipping operation stopped, the receivingControlWebService makes sure the surge tank inlet is closed.

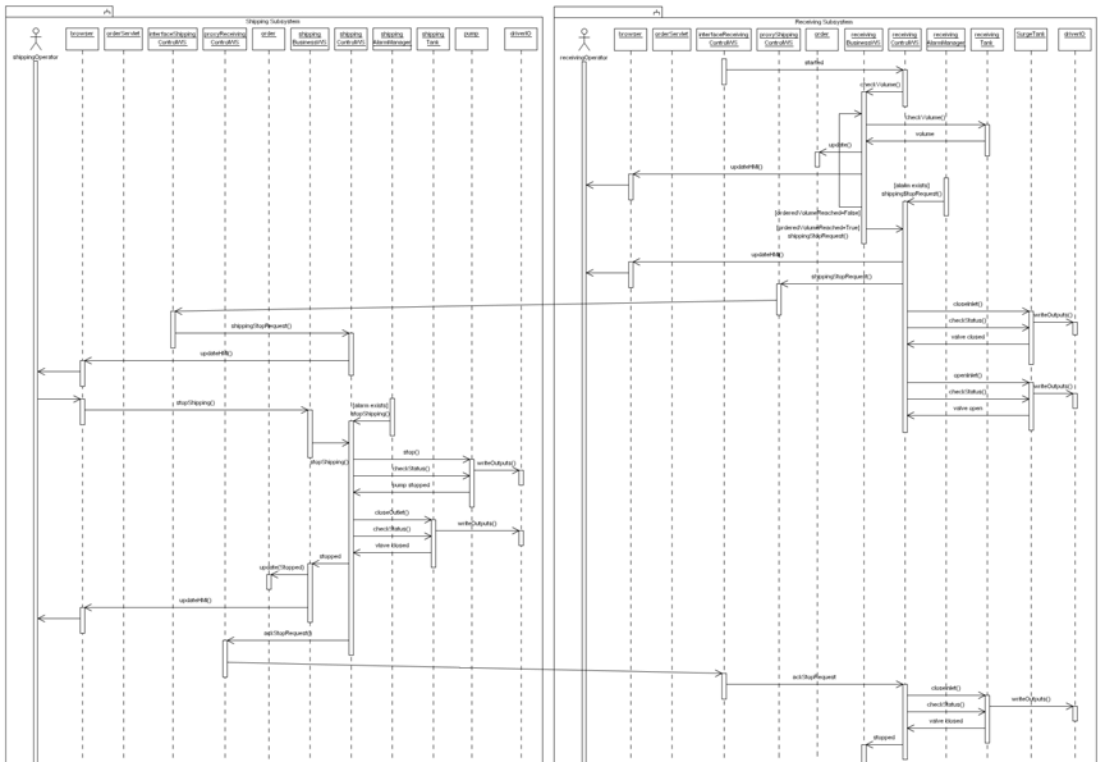


Figure 6.17: Sequence Diagram (Overall View of Stop Shipping Process)

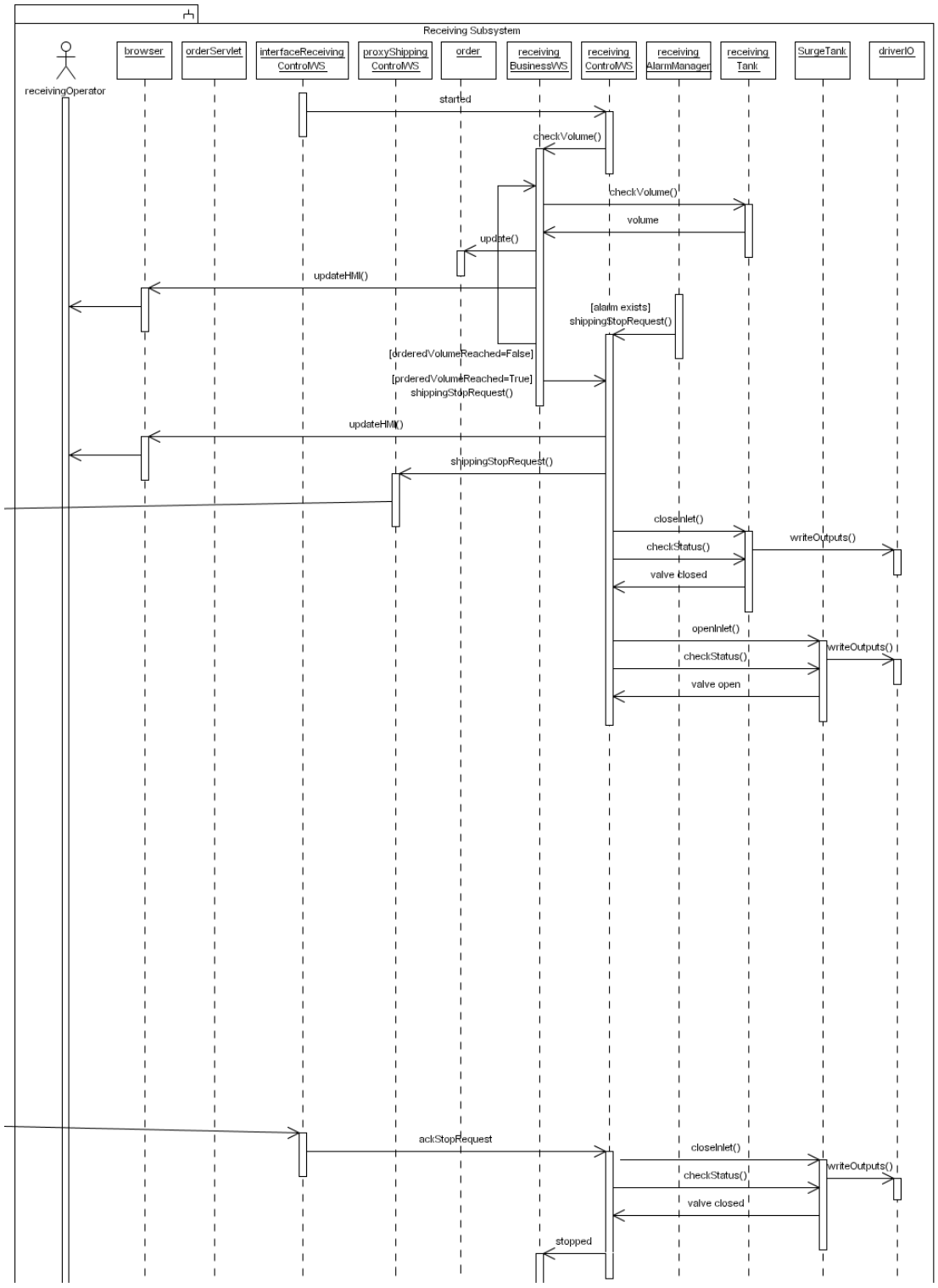


Figure 6.18: Sequence Diagram (Stop Shipping)

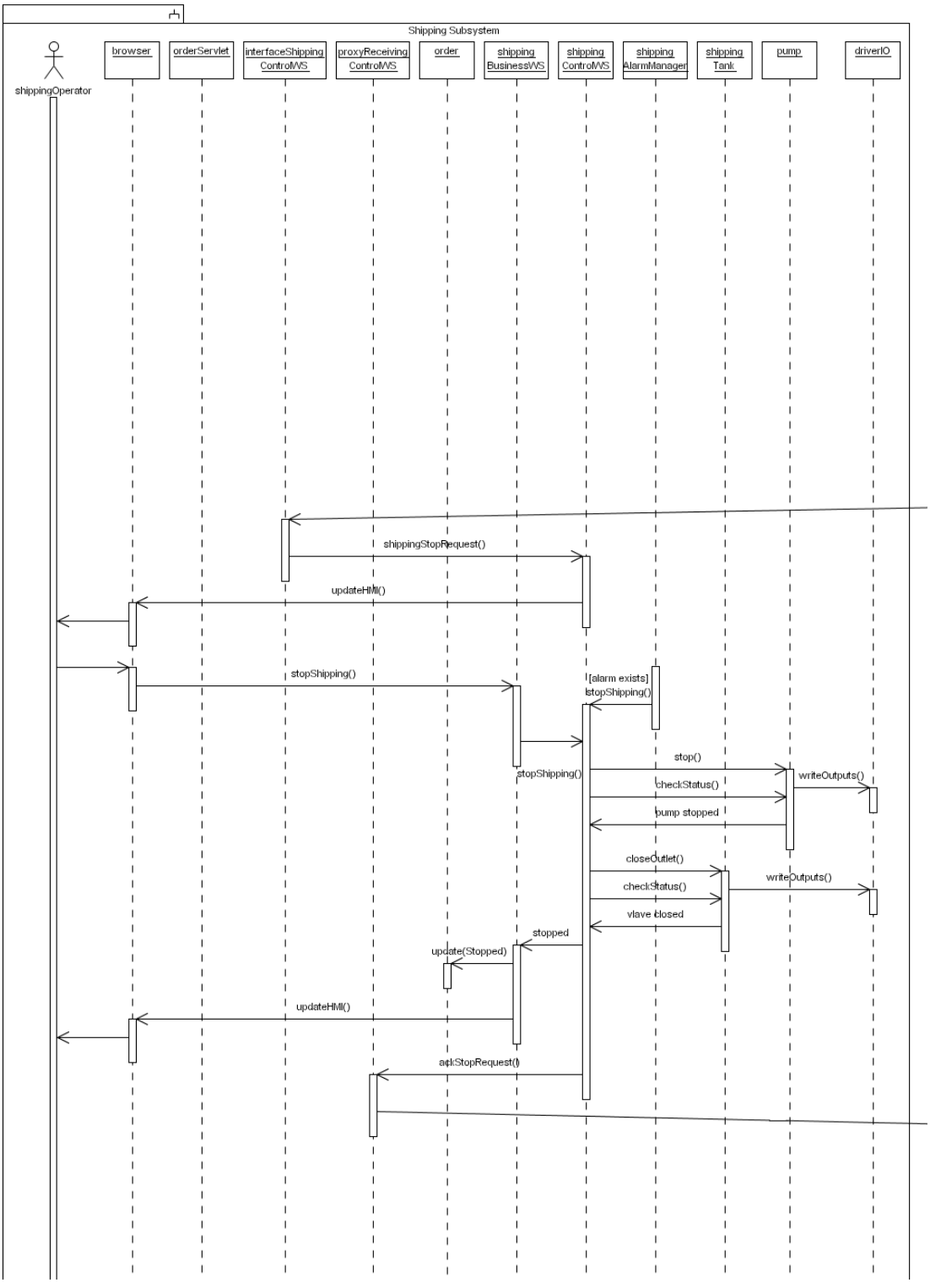


Figure 6.19: Sequence Diagram (Stop Shipping – Continued)

Close Order Process. To summarize the sequence of order closure process, the following steps are implemented:

17. If the ordered volume is completely delivered at the receiving plant, the receivingBusinesWebService updates the order as completed, marshals the order into an XML document, and sends it back to the shipping plant.
18. The shippingBusinessWebService receives the XML document, updates the order as completed, and sends the XML document to the dispatching center.
19. The dispatchingBusinessWebService receives the XML document, and updates the order as closed.

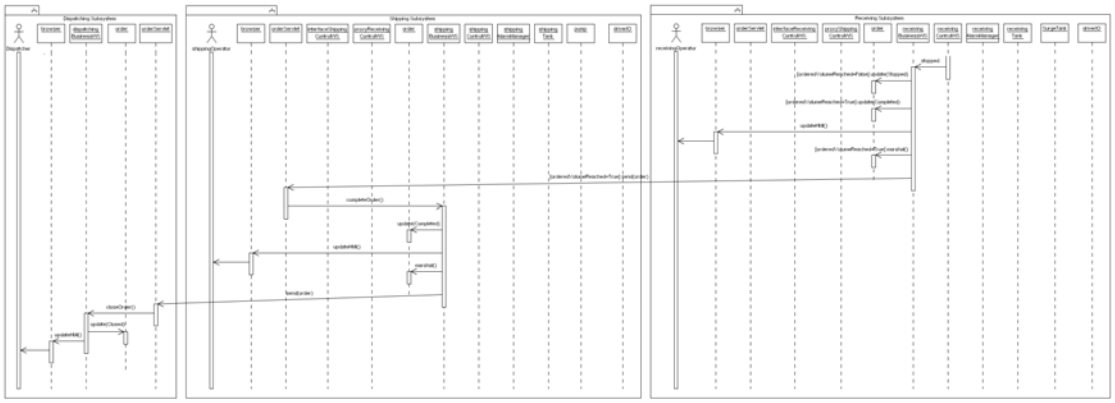


Figure 6.20: Sequence Diagram (Overall View of Close Order Process)

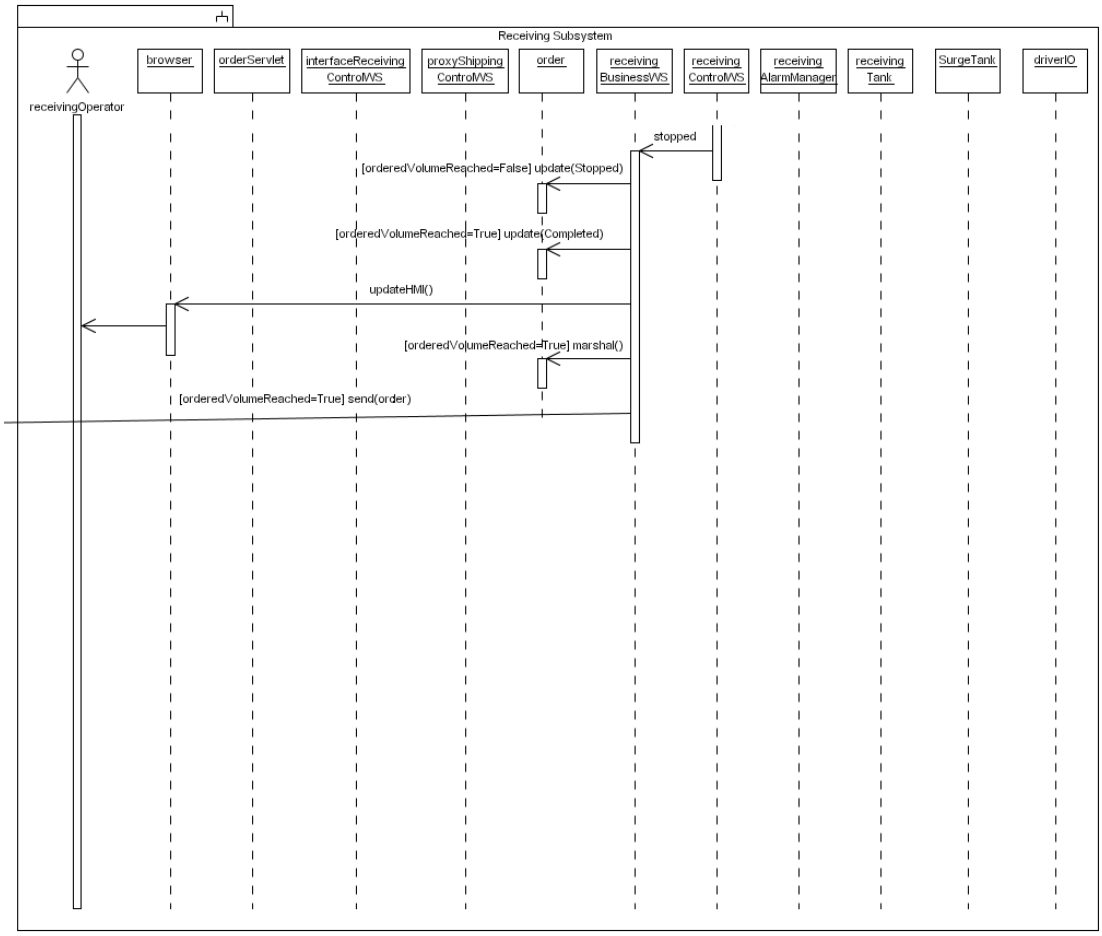


Figure 6.21: Sequence Diagram (Close Order)

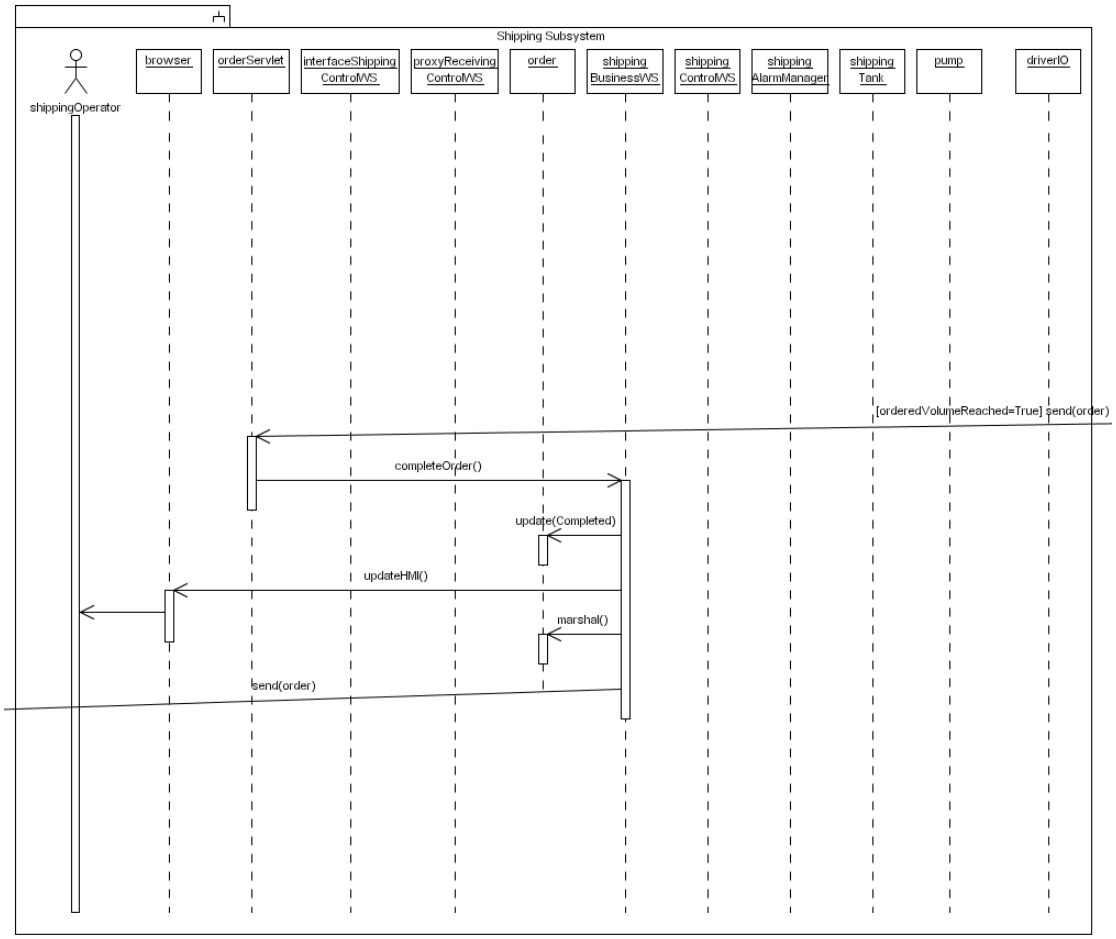


Figure 6.22: Sequence Diagram (Close Order – Continued)

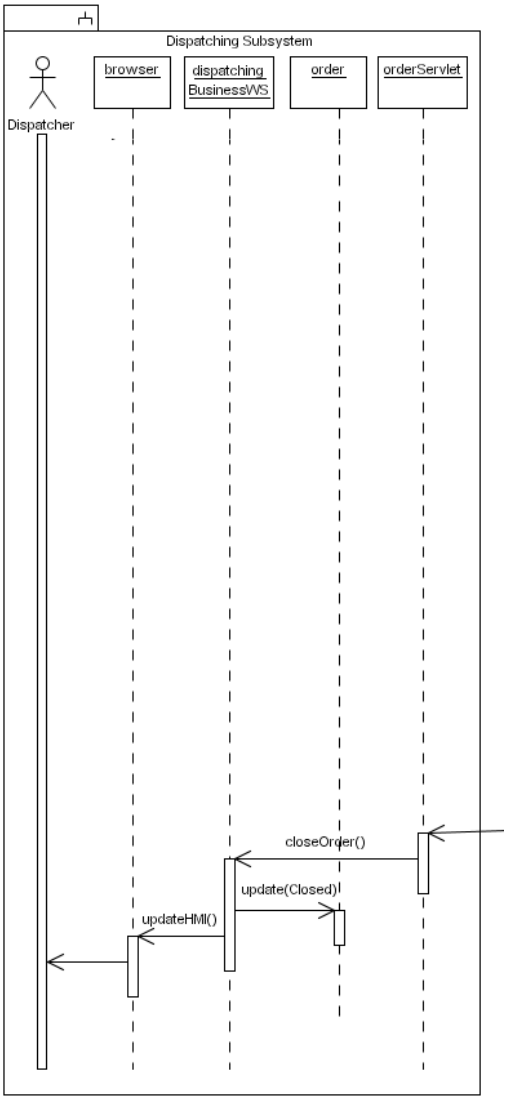


Figure 6.23: Sequence Diagram (Close Order – Continued)

6.6 State Diagram

The state diagram (Figure 6.24) shows the different states of the system and how the transitions between them take place. In this project, we assume that each of the three nodes (dispatching, shipping, and receiving) will have four states based on the shipment order status.

6.6.1 Order States

Order states at the dispatching node:

1. Created
2. DownloadedShipping
3. DownloadedReceiving
4. Closed

Order states at the shipping node:

1. Downloaded
2. Shipping
3. Stopped
4. Completed

Order states at the receiving node:

1. Downloaded
2. Receiving
3. Stopped
4. Completed

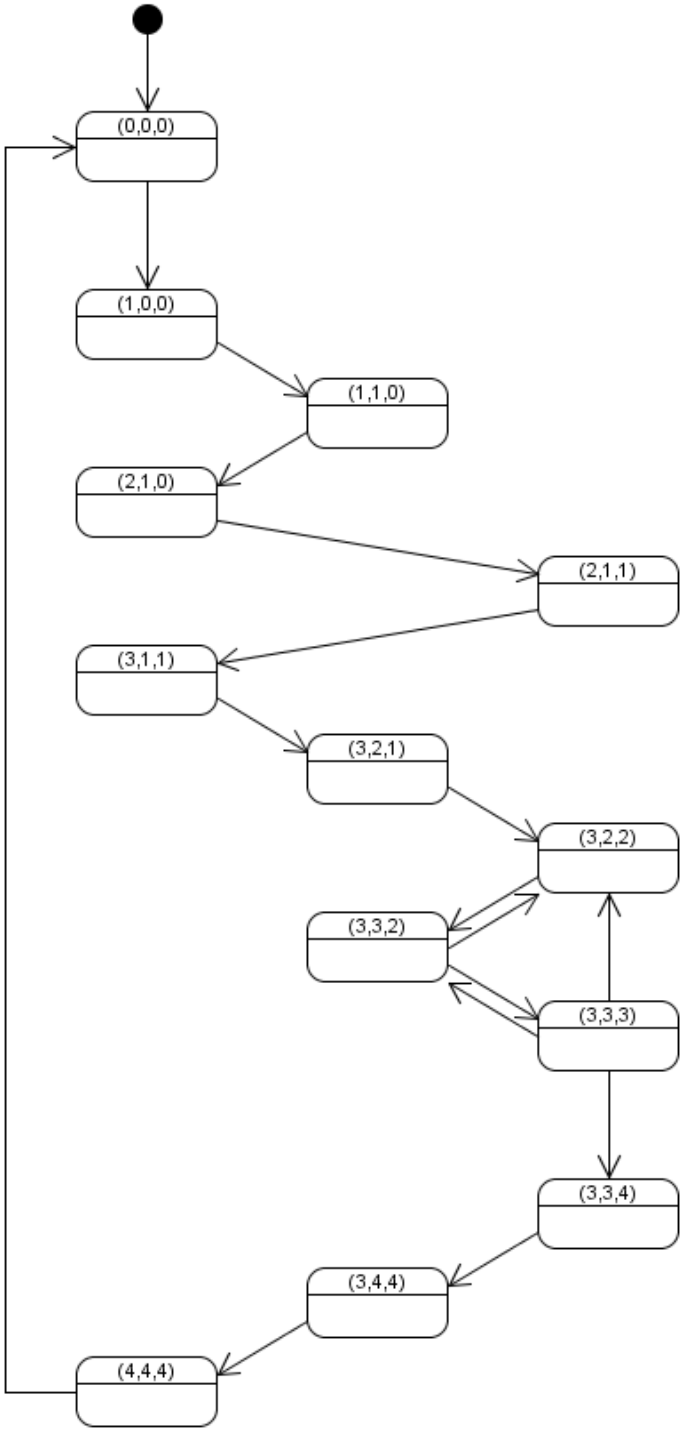


Figure 6.24: State Diagram. (Numbers between brackets indicate the order status at the dispatching, shipping, and receiving nodes, respectively.)

6.6.2 System States

The combined system states that result from these different states will be 4 * 4 * 4 = 64 states. However, under the project design, only 13 states are possible. They are described as follows:

(0,0,0) Order does not exist yet.

(1,0,0) Order created by dispatching center.

(1,1,0) Order downloaded at shipping plant, but no acknowledgement.

(2,1,0) Acknowledgement received at dispatching center.

(2,1,1) Order downloaded at receiving plant, but no acknowledgement.

(3,1,1) Acknowledgement received at dispatching center.

(3,2,1) Order started shipping.

(3,2,2) Order started receiving.

(3,3,2) Order stopped shipping (either alarm or volume completed.)

(3,3,3) Order stopped receiving (either alarm or volume completed.)

(3,3,4) Order completed at receiving plant.

(3,4,4) Order completed at shipping plant.

(4,4,4) Order closed by dispatching center.

Note that at state (3,3,2), if the shipping stopped due to an alarm, and the alarm was cleared, the operation may resume shipping again, which means going back to state (3,2,2). Similarly, at state (3,3,3), if the receiving stopped due to an alarm (either from shipping or receiving), and the alarm was cleared, the operation may resume receiving again (if receiving alarm was cleared) which means going back to state (3,3,2); or the operation may resume shipping again (if shipping alarm was cleared) which means going back to state (3,2,2).

CHAPTER 7

FINDINGS AND RESULTS

In this chapter, we summarize the findings of our investigation study and show the results of testing the research hypothesis. We then make some conclusions about the use of Internet-based SCADA system, and present some suggestions for future work.

7.1 Research Findings

From the two design projects presented in Chapters 5 and 6, we learned several things about the use of Java and XML in Internet-based SCADA systems. These key concepts then allowed us to draw conclusions about Internet-based SCADA systems and how to improve their design to meet the required applications. The following sections summarize what we could achieve by using each technology.

7.1.1 Java Uses

Platform Independence. Achieving portability is the key purpose of using the Java environment. It allows end users to select whatever hardware platform and operating system they desire. It also allows developers to market their products onto whatever hardware platform or operating system their clients use. Sun Microsystems provides three versions of the Java platform. The Java 2

Standard Edition (J2SE) is intended for desktop users, the Java 2 Enterprise Edition (J2EE) is for enterprise applications, and Java 2 Mobile Edition (J2ME) is for wireless device applications. The J2EE platform is the most suitable environment for Internet-based SCADA system as it supports development and deployment of web services.

Application Development. The Java programming language can be used to develop the server-side Java servlets and web services that are required for implementing the business and control logic, as well as to handle the data communication with other nodes over the network. Java is versatile language that covers a wide range of applications using a single programming language. Also, the Java language is expandable in that new class libraries are constantly being made available for programmers to download and incorporate into their applications.

HMI Development. The Java programming language can also be used to develop the client-side Java applets and web pages that are required for the human-machine interface (HMI). Java provides a full suite of graphical user interface (GUI) development tools, which can be used very efficiently to develop SCADA mimic displays, trends, and generate reports.

Distributed Processing. The Java environment allows developers to create and deploy distributed applications over wide area networks. The Java servlet/applet technology provides a way to distribute the data processing load between the server and client so that the application performance is optimized. This concept could benefit SCADA systems design such that system performance is always maintained.

7.1.2 XML Uses

Data Storage. The XML document provides a common medium to store process data, business information, and related system configuration, all in one standard format. This was shown in our design in Chapter 6 by storing the shipment order information in an XML document. The XML parser will check the validity of the data in this document by comparing it with the constraints defined in a DTD or schema file. This provides a mechanism to check the quality of data.

Data Interchange. Storing the information in an XML document makes it easy to share this stored information with any other machine on the network, as long as it knows the constraints set upon the information. And because XML is simply text, it can be moved between various machines regardless of their hardware platform or operating system.

Distributed Processing. The web services approach to XML implementation provides the ability to create versatile distributed applications that can communicate over wide area networks. And because the SOAP remote procedure call messages are carried over the standard HTTP message, they can pass through most firewalls without being blocked. To ensure the security of such messages, standard encryption mechanisms can be implemented during the message transmission phase to protect it against unwanted hacking.

HMI Personalization. This is a feature of XML that we did not include in our investigation, but it has been widely implemented in Internet applications. It is concerned with the activity of developing different user interfaces for different classes of users. For example, a manager display may provide summarized

process data as opposed to detailed process data used by the operator. Similarly, an engineering display may include historical process analytic data as opposed to current process data used by the operator. Customizing the display for each user has been a tedious job with previous technologies (e.g. HTML). With XML, the data is separated from the display and is stored in a structured format. The graphical display acts as a container that searches for the required information and presents only relevant data to the user.

7.2 Hypothesis Test Results

Our research hypothesis was to evaluate whether Java and XML can improve the design of current Internet-based SCADA system, and from the result we showed in the previous section, the answer is indeed "Yes." The results are summarized in Table 7.1. It is basically a comparison between the three types of SCADA systems discussed in this report, namely the traditional SCADA systems (Chapter 2), the current Internet-based SCADA systems (Chapter 3), and the proposed Internet-based SCADA system (Chapter 6).

The comparison is based on a set of qualitative criteria which were identified as part of this thesis work. These criteria are by no means exclusive and additional criteria could be included. The ones shown in Table 7.1 are merely intended to represent the major concern areas from the end user point of view. Below is a brief discussion of each.

Technology Standards. The main objective of implementing an Internet-based SCADA system is to leverage the existing technology standards and apply them for process automation. Traditional SCADA system are characterized by

TABLE 7.1: COMPARISON BETWEEN TRADITIONAL AND INTERNET-BASED SCADA SYSTEMS

	Traditional SCADA System	Current Internet-Based SCADA System	Proposed Internet-Based SCADA System
Technology Standards	Mostly proprietary	Open standards	Open standards
Networking and Communications	TCP/IP or other protocols over public (phone lines, radio) or private (company-owned) network	TCP/IP over public (Internet) or private (intranet) network	TCP/IP over public (Internet) or private (intranet) network
Data Interchange	Proprietary vendor-specific interfaces or open protocols (e.g. Modbus, OPC)	Open web protocols (e.g. HTML, HTTP)	Open web protocols (XML, SOAP/HTTP)
Network Performance	Deterministic, vendor-optimized for real-time applications	Non-deterministic, not suitable for real-time applications, inefficient data handling (HTML)	Non-deterministic, can be optimized for data streaming, efficient data handling (XML)
Application Development	C, C++, etc.	Java, Visual Basic, etc.	Java
Application Runtime	Proprietary environment, platform-dependent	Open environment, platform- dependent or independent	Open environment, platform-independent

TABLE 7.1: COMPARISON BETWEEN TRADITIONAL AND INTERNET-BASED SCADA SYSTEMS (CONTINUED)

	Traditional SCADA System	Current Internet-Based SCADA System	Proposed Internet-Based SCADA System
HMI Development	Proprietary software	Open web technologies (e.g. HTML, Java, ActiveX)	Open web technologies (e.g. HTML, XSL, Java)
HMI Personalization	Vary based on vendor development tools	Low, requires major development effort	High, data is separated from display format
HMI Performance	High, vendor-optimized for real-time applications	Low, not suitable for real-time, requires additional plug-ins, graphics files (JPEG, GIF)	High, can be optimized for data streaming, XML-based graphics (SVG)
Security	Built-in	Standard web security methods (SSL, VPN, etc.)	Standard web security methods (SSL, VPN, etc.)
Reliability	High, may vary depending on vendor technology	Low, inefficient handling of data	High, efficient handling of data, XML data validation

proprietary, vendor-specific technologies. The Internet uses internationally accepted open standards.

Networking and Communications. The Internet is mainly using the TCP/IP networking suite. This is already being implemented in many SCADA systems, however proprietary communication protocols still exist specially in older RTUs. Using a standard TCP/IP facilitates integration efforts with enterprise level system.

Data Interchange. Although there seems to be an industry trend towards using Ethernet and TCP/IP for the lower layer network protocols, the industry is still plagued with inconsistent standards for the application layer protocols. Custom interfaces are usually developed to link between the different protocols. With Internet-based SCADA systems, the upper layer protocols are based on common standards maintained by the W3C.

Network Performance. This is an area where the Internet-based SCADA system is not as capable as a traditional SCADA system. Internet communications are typically non-deterministic as the network route that the message takes is unknown. However by introducing XML, we believe that separating the data from the typical HTML format and transmitting it independently will increase overall network performance considerably.

Application Development. This area is important during the initial project design phases, as well as the subsequent support and maintenance phases. Object-oriented languages are now the standard for application development since they were designed to improve the ease of development and consistency of software applications.

Application Runtime. Implementing a pure Java application has the advantage of platform-independence. However, this has the cost of slower performance compared to other languages due to the abstraction process which happens at the virtual machine layer. Nevertheless, with faster hardware platforms nowadays, the difference in performance is becoming negligible for most practical applications.

HMI Development. The SCADA's human-machine interface is as important as its control function. Traditional SCADA systems often provide their dedicated HMI development packages. With Internet-based SCADA systems, open web technologies are used.

HMI Personalization. The proposed SCADA system design which uses XML will be easier to develop HMI in than current Internet-based SCADA systems which use HTML only. This is because of the separation of data from display.

HMI Performance. The proposed SCADA system design will also have considerable performance improvement over current Internet-based SCADA systems which use HTML only. This is because of the efficient data handling performed by XML.

Security. Internet-based SCADA systems are generally less secure than traditional SCADA systems due to their openness and remote connectivity. Traditional SCADA systems may employ security schemes built into the system at design stage, whereas the Internet-based SCADA systems utilize the standard web security methods such as encryption and tunneling. With proper implementation, the security of internet-based SCADA system can be managed.

Reliability. Internet-based SCADA systems are also less reliable than traditional SCADA system due to the non-deterministic nature of network performance on the Internet. However by introducing XML, we believe that separating the data from the typical HTML format and transmitting it independently will increase the data communication reliability.

7.3 Conclusions

Having presented our research hypothesis and the positive results we obtained, we can make some general conclusions about the design and application of Internet-based SCADA systems. These are summarized in the following four points:

- Internet-based SCADA systems can improve interoperability and can be more cost effective
- However, Internet-based SCADA systems are still less reliable and secure than traditional SCADA systems
- Java and XML can improve internet-based SCADA systems' performance, functionality, reliability and security
- Java and XML require careful modeling of the process and its constraints

7.4 Future Work

The work presented in this thesis is merely to investigate the concepts of internet-based SCADA systems, discuss their design issues and recommendations, and evaluate what applications can benefit from such

systems. The thesis demonstrated these concepts by discussing a lab-scale test system which was conducted in a separate project at the Systems Engineering labs at KFUPM, in addition to a design proposal for large-scale SCADA systems, based on the Java and XML technologies.

Further research following this thesis work shall contain actual implementation testing of the proposed design, investigation of the scalability of such systems, their performance in real environments, security concerns and considerations, field implementation issues, local regulations and policies, and so on. The scope of this thesis work cannot cover all these issues due to the limited amount of time and resources. In addition, internet technologies are evolving rapidly and the strategy of how to implement them in the process automation world requires the group effort of academia and industry.

REFERENCES

[1] Engineering Standards, *Saudi Aramco*, 2003.

[2] Engineering Manuals, *ChevronTexaco*, 2002.

[3] O. Toker and F. Al-Sunni, "Java Based Distributed Control System Over the Internet", *KACST AR20-74 Final Report, King Fahd University of Petroleum & Minerals*, Aug. 2003.

[4] J. Sanchez et al., "Virtual and Remote Control Labs Using Java: A Qualitative Approach", *IEEE Control Systems Magazine*, Apr. 2002.

[5] V. Ramakrishnan et al., "Development of a Web-Based Control Experiment for a Coupled Tank Apparatus", *Proceedings of the American Control Conference*, Jun. 2000.

[6] Extensible Markup Language, *http://www.w3c.org/xml*. 2004.

[7] J. Fulcher, "Hit the Floor Running: Java Technology-based Controls Promote Shop-floor Efficiency, Offer Enterprise Architecture", *Open Manufacturing Journal*, Fall 1998.

[8] S. Hill, Jr., "Java Computing Offers manufacturers Greater Control: Development of Java Technology Devices Heralds a New Paradigm", *Open Manufacturing Journal*, Summer 1998.

[9] J. Morgenthal, "Portable Data/Portable Code: XML & Java Technologies", *White paper prepared by NC.Focus for Sun Microsystems, Inc.*, May 2000.

[10] C. Vawter and E. Roman, "J2EE vs. Microsoft .NET: A Comparison of Building XML-based Web Services", *White paper prepared by the Middleware Company for Sun Microsystems, Inc.*, Jun. 2001.

[11] P. Pinceti, "How Will XML Impact Industrial Automation?" *ISA InTech*, Jun. 2002.

[12] J. Bono, "XML Reaches Factory Floor's Automation Islands", *ISA Industrial Computing*, Aug. 2000.

[13] C. Gunst and J. Stein, "The Internet: Fast, Cost Effective Methods to Improve Communications on Your Plant Floor", *Intellution, Inc.*, 1997.

[14] J. Kennedy, "Internet/Intranet and Object Technology for the Process Industries", *OSI Software, Inc.*, 1997.

[15] R. Eisele, “Transforming Internet Technology into Manufacturing Solutions in the Chemical Industry: An Update on the Industrial Desktop”, *OSI Software Seminar*, 2001.

[16] J. Patrick Kennedy, “Industrial Desktop - What, Why, How?” *Chemical Engineering Magazine*, Jan. 1996.

[17] R. Heersink and S. Wright, “WWW.WhereIsTheValue.Com?” *Hydrocarbon Processing*, May 2000.

[18] R. Heersink and S. Wright, “Secure Management of Manufacturing Operations Through the Power of the Web”, *National Petrochemical & Refiners Association Computing Conference*, Nov. 2000.

[19] S. Diehl and R. Moyes, “Fueling the Midstream Enterprise with Internet Enabled SCADA”, *Matrikon, Inc.*, Dec. 2000.

[20] R. Bailey, “Internet-based SCADA: Evaluate Your Options Carefully”, *Instrumentation & Control Systems*, Jul. 2000.

[21] “Supervisory Control and Data Acquisition (SCADA): The Challenge of Increased Computer Power, Higher Speeds and Modern Networks”, *Data Comm for Business, Inc.*, Oct. 1999.

[22] S. Boyer, “SCADA: An Introduction Including What Not to SCADA”, *ISA Encyclopedia of Measurement and Control*, vol. EMC 37.01, 2001.

[23] “SCADA Systems Worldwide Outlook”, *ARC Advisory Group*, 2000.

[24] “SCADA Systems for Electric Power Worldwide Outlook”, *ARC Advisory Group*, 2001.

[25] R. Cottingham, “Wireless Web Accesses Wellsite SCADA”, *Oil & Gas Journal*, 10 Dec. 2001.

[26] G. Veitch et al., “Field Trial Tests Web-Based Wireless eSCADA”, *Oil & Gas Journal*, 16 Sep. 2002.

[27] H. Kusch, “Use of Portal Technology for Business Impact: CommonView Refining Operations Portal”, *Indx User Group Meeting*, 2002.

[28] R. Fan, “Process Systems Integration Over the Internet/Intranet”, *KFUPM Workshop on Information and Computer Science*, Mar. 2002.

[29] Java Language, *http://java.sun.com*. 2004

[30] J. Bosak, “XML, Java, and the Future of the Web”, *Sun Microsystems, Inc.*, Mar. 1997.

[31] P. Whitehead et al., *Java and XML*, Wiley Publishing, New York, NY, 2002.

[32] N. Chase, *XML and Java from Scratch*, Que Publishing, Indianapolis, IN, 2001.

[33] B. McLaughlin, *Java and XML*, O'Reilly & Associates, Sebastopol, CA, 2001.

[34] B. McLaughlin, *Java and XML Data Binding*, O'Reilly & Associates, Sebastopol, CA, 2002.

[35] R. Wolter, "XML Web Services Basics", *Microsoft*, Dec. 2001.

[36] A. Walsh, *J2EE 1.4 Essentials*, Wiley Publishing, New York, NY, 2003.

[37] D. Chappell, Understanding .NET, Addison Wesley, Indianapolis, IN, 2002.

[38] S. Weygandt and D. Hardin, ".NET Industrial Automation", *ISA Conference*, 2002.

[39] D. Harrold, "XML Delivers", *Manufacturing*, Oct. 2003.

[40] M. Brooks, "The Affects of E-business on Refining and Petrochemical Plant Operations", *National Petrochemical & Refiners Association Computer Conference*, Nov. 2000.

[41] Y. Jung, "A Real-time Plant Database Based on XML and its Application to a Computer Based Procedure", *ISA Conference*, 2001.

[42] Unified Modeling Language, *http://www.uml.org*. 2004.

[43] A. Holub, "UML Quick Reference", *http://www.holub.com*, 2004.

[44] D. Braun et al., Unified Modeling Language (UML) Tutorial, Kennesaw State University, 2001.

VITA

- Ramadhan Alaaddin Nouraddin Fan.
- P.O. Box 5235 Dhahran 31311 Saudi Arabia.

  Email: ramadan.fan@aramco.com

- Born in Jeddah, Saudi Arabia in September 29, 1974.
- Received Bachelor of Science (B.Sc.) degree in Systems Engineering from King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia in July 1997.
- Joined Saudi Arabian Oil Company (Saudi Aramco), Dhahran, Saudi Arabia in August 1997.
- Worked for ChevronTexaco Corporation, San Ramon, California, USA from January till June 2003.
- Received Master of Science (M.Sc.) degree in Systems Engineering from King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia in June 2004.