Improving Internet-Based SCADA Systems Using Java and XML

Ramadan A. Fan
Saudi Aramco
ramadan.fan@aramco.com

Dr. Lahouari Cheded and Dr. Onur Toker
Department of Systems Engineering, KFUPM
cheded@ccse.kfupm.edu.sa, onur@ccse.kfupm.edu.sa

Abstract

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. 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. In this paper, we propose a design for an Internet-based Supervisory Control and Data Acquisition (SCADA) system using Java and XML. SCADA systems are typically distributed in nature and are more suitable for this investigation than other types of control systems.

1 Introduction

The use of the Internet in process control and automation industry has been driven by two main forces: (1) commercial trends in computer and information technology fields, and (2) the needs of process and manufacturing industries. Whenever a match occurs between these two driving forces, companies try to capitalize on 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 process control systems, such as Distributed Control Systems (DCS), Supervisory Control and Data Acquisition (SCADA) Systems, and Programmable Logic Controller (PLC) Systems.

Each of these control systems has been traditionally designed for a specific purpose, depending on the controlled process at hand. For example, the PLC system is used to control small to medium plants where the controlled process is mainly sequential and digital in nature. The DCS system is used to control large plants where the process involves many continuous control loops. The SCADA system is used to control applications that require communication with geographically remote areas, such as pipelines or power distribution facilities. The distinction between these systems, however, is becoming blurry as they begin to adopt each other's functionality and use the latest advancements in computer and networking technologies.

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. 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.

The Internet, however, has its limitations when compared to a traditional control system. For example, Internet applications are typically based on textual web pages and transactional databases; whereas control systems require interactive graphical displays with real-time dynamic data. These 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. Nevertheless, recent emerging web technologies are promising to overcome many of these limitations.

2 Evolution of SCADA Systems

Supervisory Control and Data Acquisition (SCADA) system is an industrial measurement and control system that enables a plant operator to monitor one or more distant facilities and send limited control instructions to those facilities [1]. Figure 1 shows a typical SCADA system architecture. It is similar to DCS systems in that they are both used for process control and monitoring. However, they differ in that DCS systems monitor and control units that are usually located within a confined area and the communications are usually fast and reliable, while SCADA systems cover larger geographic areas and rely on a variety of communications systems that are normally less reliable.

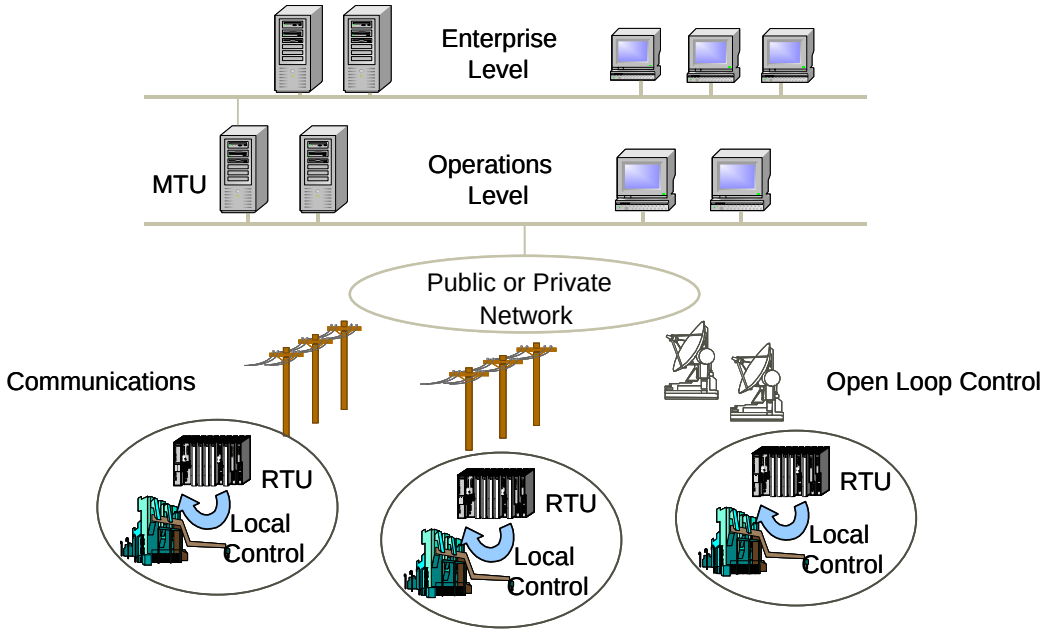

Figure 1. Typical SCADA system

SCADA systems also use mostly open loop control due to the less reliable communication, while DCS systems employ significant amounts of closed loop control. The HMI interface 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 or analog values such as numbers, bars or charts. Control elements are shown as buttons or setpoint values.

A typical SCADA system consists of four main components:

- Field Instrumentation: This includes 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.

- Remote Terminal Units (RTU): These are the controllers that are interfaced to the field instrumentation. An RTU can be single board or modular unit. The single board provides a fixed number of input/output (I/O) interfaces. The modular type is an expandable remote station and more expensive than the single board unit. Programmable Logic Controllers (PLCs) are sometimes used with SCADA systems instead of RTUs due to their flexibility and programmability.

- Master Terminal Unit (MTU): This is the master station or host computer which acts as the centralized controller for the SCADA system. The MTU can be a single computer or a network of servers and 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.

- Communications: This is the spine of SCADA technology. In order for the central MTU to communicate with the RTUs that are located at distant locations or with the various systems that are located within the corporate network, a communication link must exist to transfer data from one location to another. There are two types of communication media: wired (electrical or fiber optic cable) and wireless (radio frequency). In most cases, a combination of more than one media is used.

Communications play a major role in SCADA systems 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.

- Field communications have been traditionally based on serial communication, but this is changing with the introduction of Fieldbus and Ethernet networking technologies that leverage the more intelligent devices, allowing improved access to real-time measurements and instrument diagnostics. Also, as SCADA systems grow 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.

- High-level supervisory systems integration is usually based on Ethernet and TCP/IP which are gaining popularity in the industry. At the host level in the central control room, Ethernet brings the data to a server that provides the data to workstations for operation, engineering, and maintenance. The database applications are usually made compatible with the web, so that information can be dispensed to the corporate intranet or the Internet.

Top

Although common networking standards like Ethernet and TCP/IP have become the de facto standards in industrial automation, they only satisfy the lower four layers of the OSI/ISO 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. Each vendor of automation equipment has implemented its own application layer protocol (Figure 2). As a result, a standard application layer and common object model do not exist.

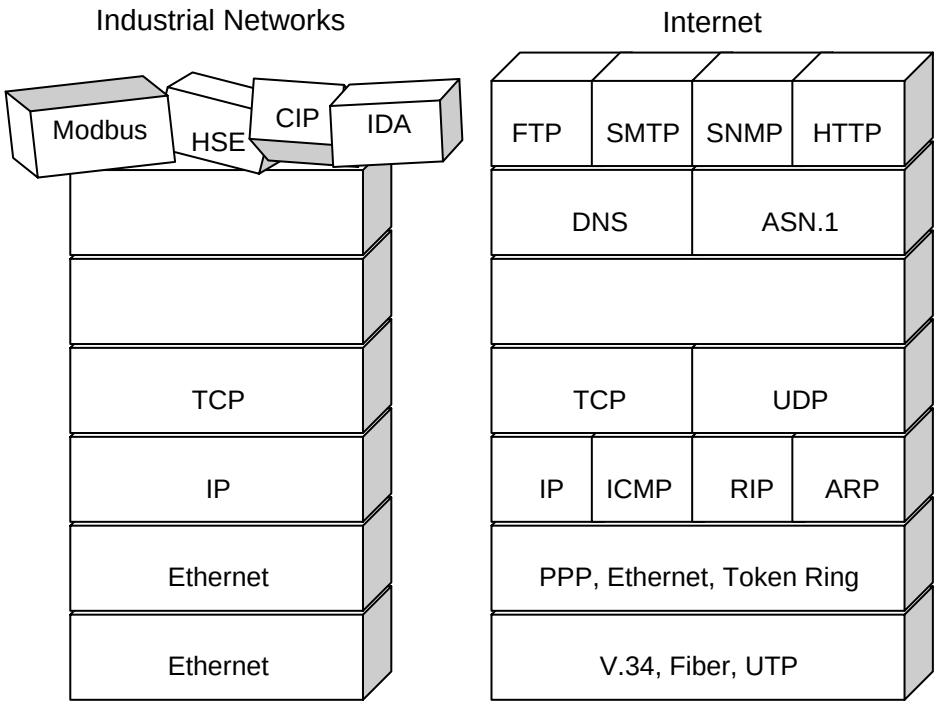

Figure 2. OSI/ISO network model for typical industrial networks and for the Internet

Internet-based SCADA systems have been recently introduced 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 using these technologies is to utilize internationally accepted standards to achieve the monitoring and control functions that a traditional SCADA system provides but with a lower cost. This usually results in better interoperability between different system components, easier dissemination of information to various applications and external systems, and unification of the HMI interface through the standard web browser. [2, 3, 4]

3 Using Java and XML

Java is the most common programming language for the Internet. 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."

Top

The Extensible Markup Language (XML) is a method for structuring and describing information. It is a subset from the Standard Generalized Markup Language (SGML), which is a specification developed 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 [5]. 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 [6, 7, 8, 9]. In addition, since W3C manages the XML specification and Sun controls the Java specification, both languages are well-defined. This leads to longer lifetime for any Java applications and any information stored with XML. Java and XML binding can be achieved through different application programming interfaces (APIs):

- Low-Level APIs are the most basic way of accessing XML information from within a Java program. 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 communication messaging. Examples of low-level APIs include the Simple API for XML (SAX), the Document Object Model (DOM), and the Java API for XML Parsing (JAXP.) [10]

- High-Level APIs hide 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 when solving specific business problems, and has become quite popular with the developers. Different approaches have been used for high-level APIs. 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 data binding. This usually consists of four steps: binding schema, class generation, unmarshalling, and marshaling. [10]

One important area where Java and XML are being implemented is in Web Services. The term Web Service is 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. [11, 12, 13, 14]

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

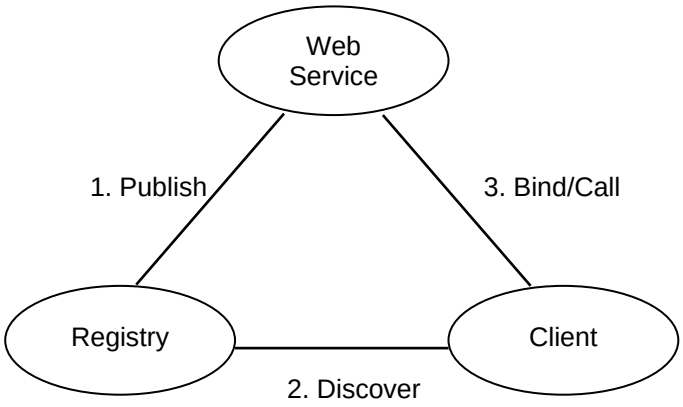

Figure 3. OSI/ISO network model for typical industrial networks and for the Internet

4 Proposed Internet-Based SCADA System

In our proposed design, we only consider the third step of the web services model, which is the data binding process. This is needed to achieve the desired integration between the various components of the SCADA system. In this section, we provide a high-level design model of our proposed system in standard Unified Modeling Language (UML) notation. The design is demonstrated using a hypothetical industrial application, namely the liquid transfer process between two tanks (Figure 4). This situation can be found in many industrial applications.

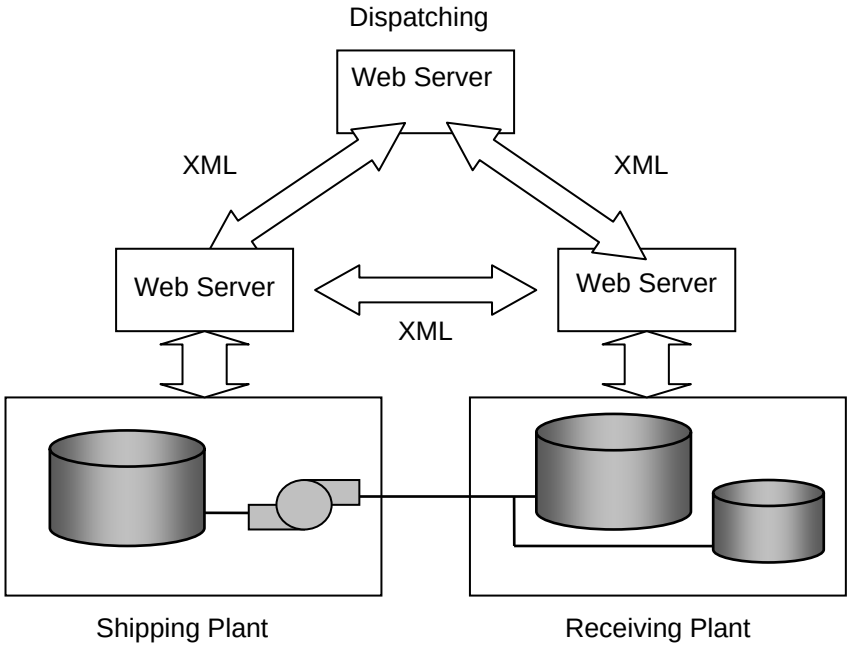

Figure 4. Overview of the tank transfer process

Figure 5 shows the use case diagram which consists of the shipping plant, the receiving plant, and the dispatching center. 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. Basically, each of the three locations will be implemented as a web server. The dispatching node can be thought of as an MTU, while the shipping and receiving nodes can be thought of as the RTUs.

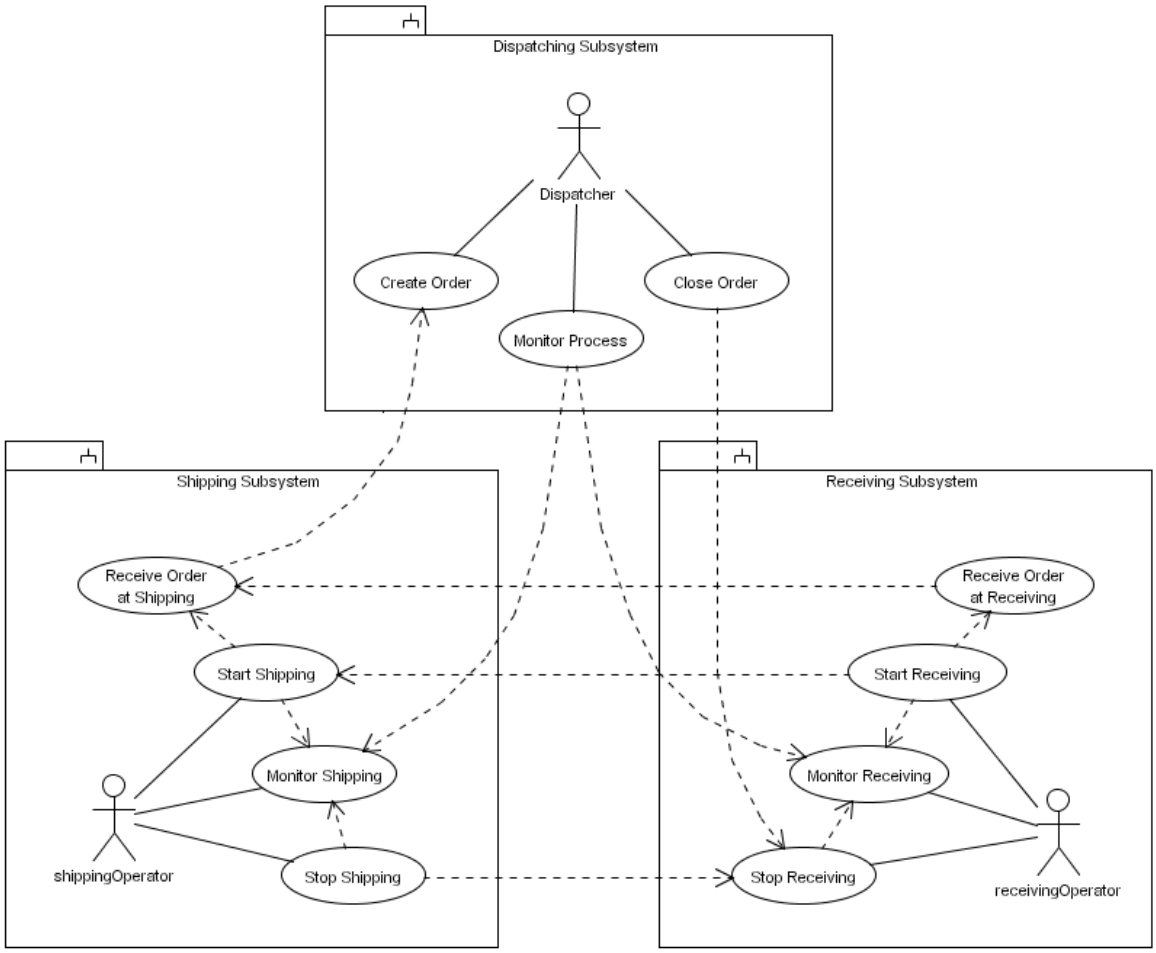

Figure 5. Use case diagram

The use case diagram also 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 once the ordered volume is completed.

The component/deployment diagram (Figure 6) 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.

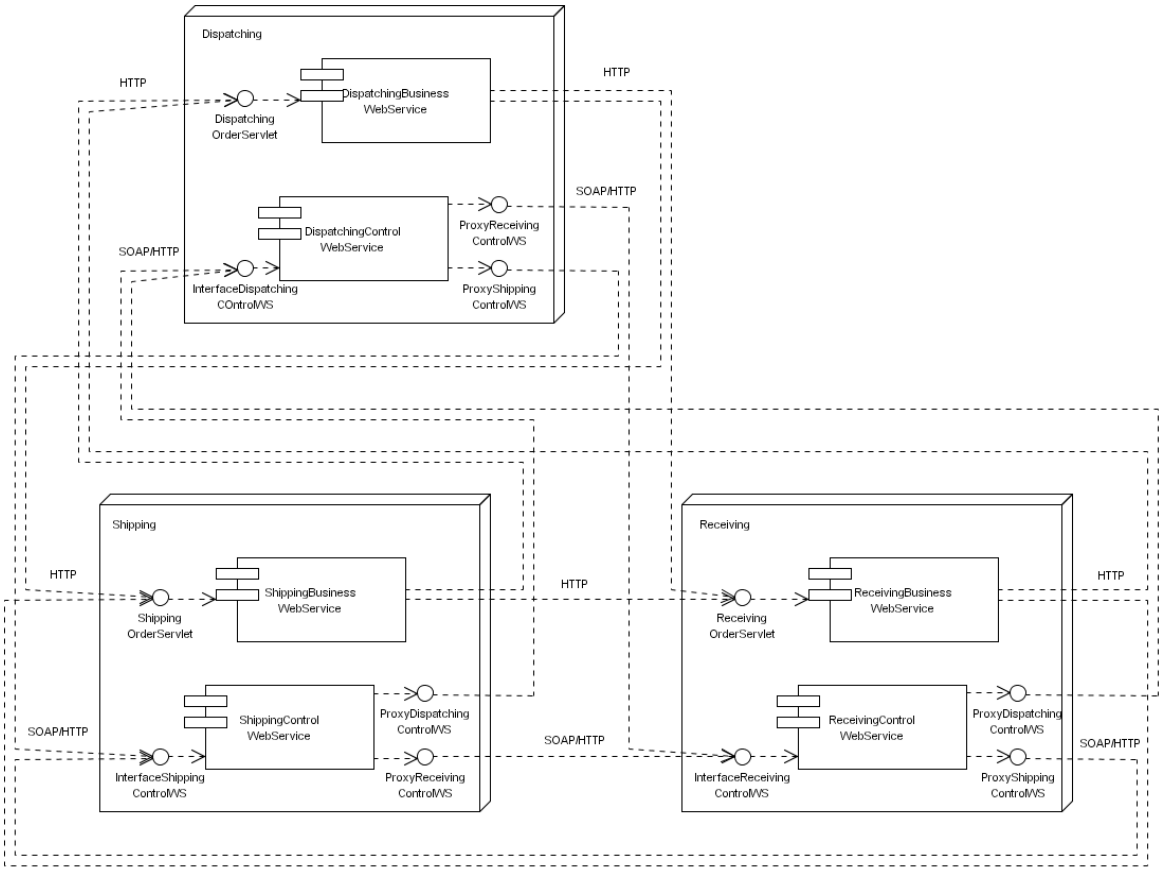

Figure 6. Component/deployment diagram

Each of the three nodes has both business and control functionalities. These are designed as business and control 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. 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.

The class diagram (Figure 7) 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 (Tank, Pump, Order), the Java drivers (to interact with the physical process equipment), the Java servlets (Business Web Service, Control Web Service, Alarm Manager), and the Java Applets (for human-machine interface). The web services concept is used for modeling Java applications that need to interact with the other remote applications, while the Java applications that run locally on the machine and do not require interaction with external applications are modeled as normal Java applications.

Top

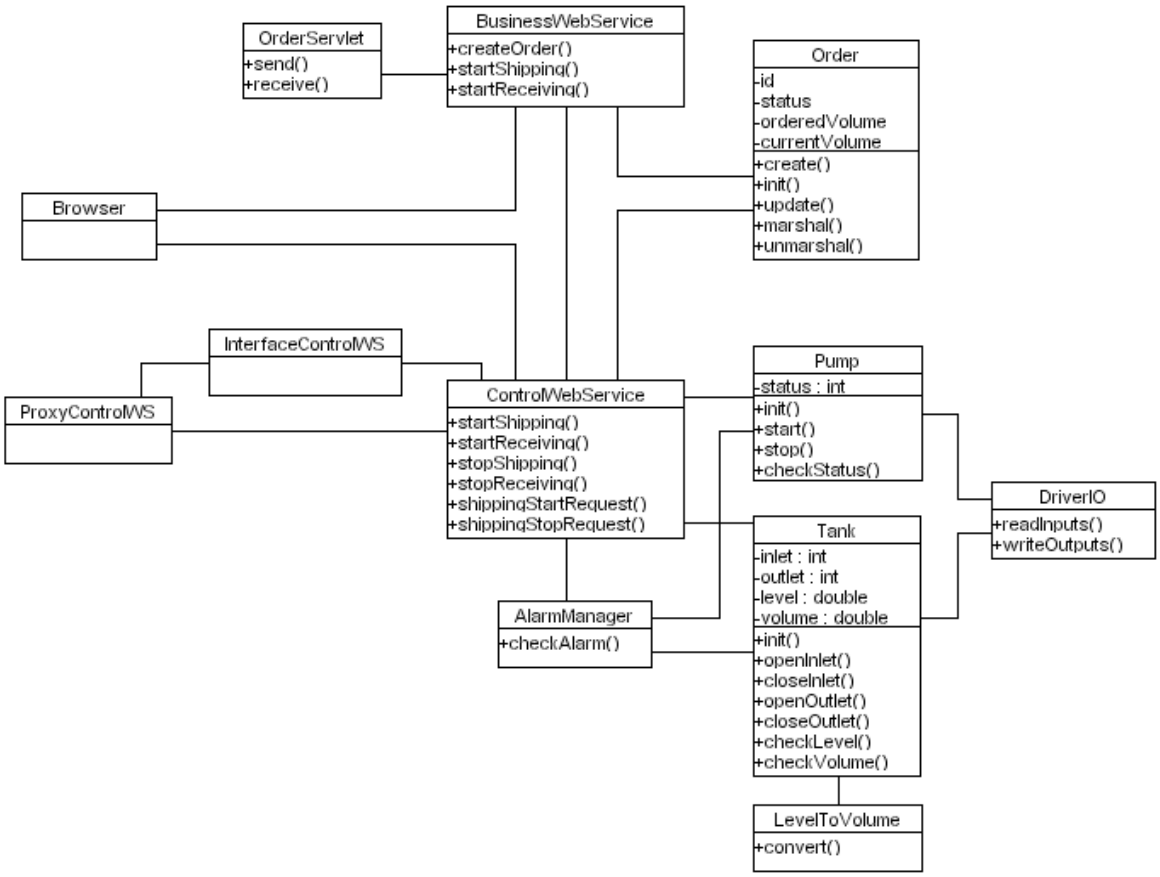

Figure 7. Class diagram

5 Conclusion

The results of our research are summarized in Table 1. It is basically a comparison between the traditional SCADA system and the Internet-based SCADA systems (including current and proposed). The comparison is based on a set of qualitative criteria which represent major concern areas from the end user point of view. These criteria are by no means exclusive and additional criteria could be added. The objective is to 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 Internet-based control systems, identify their advantages and disadvantages, and eventually select the control solution that is most suitable for their needs.

The work presented in this paper is merely to investigate the concepts of Internet-based SCADA systems. Further research 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, and local regulations and policies. The 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.

Top

	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
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

Table 1. Comparison between traditional and Internet-based SCADA systems

Top

Acknowledgment

The authors would like to acknowledge the support of Saudi Aramco and King Fahd University of Petroleum & Minerals for the development of this work.

References

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

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

[3] P. Pinceti, “How Will XML Impact Industrial Automation?” *ISA InTech*, Jun. 2002.

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

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

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

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

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

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

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

[11] R. Wolter, “XML Web Services Basics”, *Microsoft*, Dec. 2001.

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

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

[14] S. Weygandt and D. Hardin, “.NET Industrial Automation”, *ISA Conference*, 2002.

Top