A Hybrid Architecture for Mission–Critical, Real–Time Web Client Applications

EFREN CORONEL, ILSE LEAL
Instituto de Investigaciones Electricas
Gestion Integral de Procesos
Reforma 113 Col. Palmira, Cuernavaca Mor.
MEXICO

Abstract: This paper describes a hybrid software architecture that can be used to develop powerful, real–time Web applications for monitoring industrial processes with a high degree of reliability and availability. Our architecture enables developers to combine traditional desktop software development techniques with new Web development methodologies in order to achieve multiplatform, mission–critical applications that are visually attractive, flexible and modern.

Key–Words: Web Application, Real-Time, Process Monitoring

1 Introduction

In practice, mission-critical monitoring systems are developed using proprietary software with highly specialized code. These systems usually communicate with data acquisition modules or measurement instruments using different proprietary protocols like ModBus [1], DNP3 [2], OPC [3], IEC 61850 [4], etc. Therefore, these systems are platform–dependent. Since these systems are used to supervise and monitor critical processes, they must operate with a high level of reliability and availability.

Over the years, as technology has advanced, monitoring systems have become obsolete. Hence, organizations face the need to migrate their systems, which implies large investments and operational risks.

The hybrid architecture that we describe in this paper, can be used to develop mission–critical, modern Web applications that are capable of acquiring data in real time from different data sources using a wide range of specialized protocols. Therefore, systems based on our architecture can offer reliable data and high availability. By incorporating modern Web technologies, our architecture enables the development of responsive, attractive and flexible human machine interfaces that can be used in different devices. In addition, because our architecture supports multiplatform development, the resulting systems can operate in different operating systems such as Windows, Linux and OS X. This helps reduce development and deployment costs.

2 Hybrid architecture

To understand our concept of hybrid architecture, we start with a conceptual model which is based on the fact that critical process monitoring systems were developed as platform–dependent desktop applications. With this, developers seek to ensure that resulting systems are fast and robust enough to operate reliably for extended periods of time. This kind of systems offer real–time communication mechanisms with excellent performance and unmatched multiprocessing capacity.

However, over the years, new application development technologies have emerged and progressed rapidly. These new technologies are mainly oriented to the creation of modern Web systems, which can offer superior flexibility to design highly functional human machine interfaces that can be deployed in a wide variety of devices: from desktops to smart phones.

In this paper, we propose a Web architecture that combines the best of both worlds: the robustness, availability and real–time responsiveness of traditional desktop systems with the flexibility and multiplatform capacity of modern Web systems.

2.1 Mission-critical systems requirements

By definition, mission critical systems have substantial impact on an organization's processes and operations. For critical industrial process, it is important to have highly effective and reliable monitoring systems that reduce or eliminate operational risks that could cause service shortages, financial losses or security

risks. The following are some of the most important requirements in a mission–critical system:

- High degree of reliability and availability
- Low latency in communication and the capacity to acquire data from several data sources
- Ease of use
- Correct and timely presentation of information
- Secure access to critical data

In light of the foregoing requirements, the reason why most mission–critical systems use specialized software that runs on well–known, robust platforms becomes clear [5]. Still, only a few companies develop this kind of systems. Therefore, their software licenses are usually very expensive.

Web applications, developed with the most recent open standards, offer the advantage of minimizing the cost of software licenses while making it possible to develop attractive user interfaces. Nevertheless, their main disadvantage is the lack of reliable and low–latency communication and data acquisition mechanisms.

2.2 Why use Web Technologies?

Web application development has been on the upswing mainly due to great advances in consumer devices as well as new hardware technologies that are more efficient and powerful. Additionally, new software development standards improve user productivity because of their flexibility.

One of the main advantages of Web applications is their ease of development because of the wide range of standards that can be used. HTML5 offers an ideal set of features that facilitates the development of intuitive and attractive user interfaces. CSS3 provides great flexibility by making it possible to easily define and modify the appearance of a Web page. Another powerful standard is SVG (Scalable Vector Graphics), which makes it possible to specify two–dimensional vector images in XML format.

In addition, JavaScript makes it possible to access HTML, CSS3 and SVG elements through objects, which are instantiated in memory. Therefore, with JavaScript, developers can directly modify element behaviors, styles and shapes within Web applications. This enables Web developers to generate rich, dynamic content.

However, traditional Web applications are not usually used to develop mission–critical applications due to certain disadvantages regarding the way they operate internally. One of the most important factors is that, in this type of applications, there is only

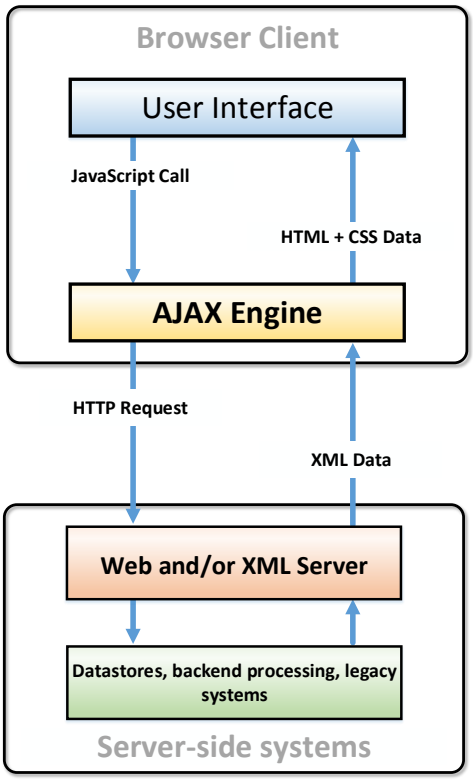

Figure 1: Tradicional Web Application

one main execution thread. Therefore, multiprocessing is not possible. Even though developers can use timers and event handlers that can be launched asynchronously, these elements are still executed in the main thread cycle. This can lead to problems that cause the application to become intermittently unresponsive.

Finally, by considering advantages and disadvantages of both architectures, desktop and Web, we have created a hybrid architecture that combines the best of both, managing to meet the requirements described in section 2.1.

2.3 Main concept

As we have mentioned, the key idea in our architecture is to integrate high performance, low–latency and high availability communication mechanisms, which are available in traditional desktop systems, with the advantages of flexible, multiplatform, modern Web applications that can operate offline, without the need for a Web server. This is possible because in our architecture, application elements are loaded only once. Therefore, our applications have all the information they need to operate and do not need to query a Web

server since objects can be accessed directly in memory. We have achieved this by designing a layered architecture that combines Java and Javascript along with other standards such as HTML5, CSS3 and SVG. Figure 2 shows the various modules that make up our architecture.

Java handles the core modules of the application. These modules communicate with JavaScript in order to link communication and data processing tasks. Presentation tasks, such as the display of information, are executed in the Web browser.

As we mentioned before, in a traditional Web application, there is only one main thread, which runs on a browser and can be accessed by JavaScript. By implementing core modules in Java, we can incorporate Java threads and achieve true multiprocessing capabilities. In order to integrate Java capabilities to Web applications, our architecture enables information exchange between Java and JavaScript. With this methodology, data acquisition from legacy systems, databases or Web services can be performed directly in Java by using asynchronous execution threads. Once acquired data are in memory, they can be accessed through JavaScript and presented in real–time to users. This mechanism is explained in more detail in section 4.1.

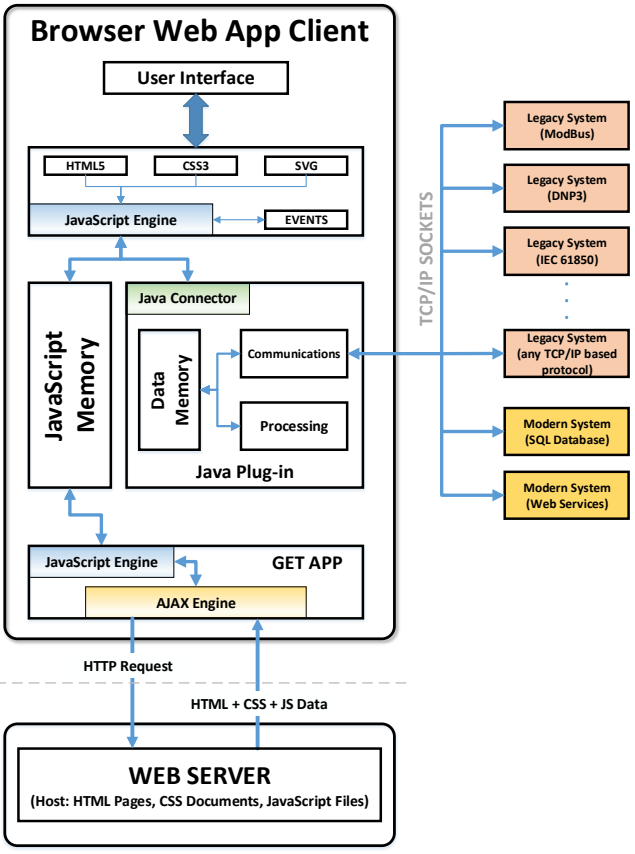

Figure 2: Main Concept

3 Solving Traditional Web Systems Limitations

One of the most important elements in any monitoring system is the possibility to connect to different data sources. This enables information exchange between measurement devices or data servers and the system. This is usually done by implementing different communication protocols. In traditional Web applications, there are very few technologies that can be used to communicate with data acquisition devices which use protocols based on TCP/IP such as ModBus, DNP3, RTP, etc. The same applies for SQL connectors or Web services, which are done with server–side technologies. In our architecture, information requests are performed directly by the client.

Another important limitation is multiprocessing capacity, which is essential for real–time systems that must operate asynchronously in order to be able to perform several tasks in parallel.

3.1 Real-time data acquisition

Real–time systems are defined as computer systems that are subject to time constraints and therefore must respond in a certain amount of time. For this kind of systems, correctness is not enough by itself: tasks must execute in a certain period of time [10]. There are two types of real–time systems: *hard* real–time systems, in which time constraints are very strict and *soft* real–time systems, which can tolerate occasional delays.

The correct implementation of mission–critical systems, such as industrial monitoring systems, using Web technologies is particularly complicated, mainly because these technologies were not designed with high–demand systems in mind. New technologies are emerging to fill this gap, but they have limitations.

One example is Websockets [6], which enables the use of sockets in Web applications. However, this technology is still at an early stage of development and has some major limitations. One of the most important is that developers cannot implement proprietary TCP/IP protocols. This is due to the fact that, in order to develop applications, developers need access to a Websockets server.

By using Java as the core of an application, developers can take advantage of a considerable set of software libraries that allow applications to connect to different data sources through native TCP/IP sockets. With this, proprietary protocols can be implemented

just as if Web applications were traditional, platform–dependent desktop applications. Therefore, applications developed using our architecture can communicate with both legacy and modern servers.

3.2 Historical data acquisition

In section 3.1 we mentioned real–time data acquisition. However, another key feature in our hybrid architecture is the capacity to acquire historical data. To achieve this, evidently, it is necessary to have access to servers that store and provide this information. With historical data, systems can provide reports or trend graphs that illustrate the state of monitored processes.

Java's extensibility makes it possible to access Web services, perform SQL queries or communicate with OPC or PI (OSIsoft) servers. In Java, this information can be obtained through a dedicated acquisition thread or can be triggered on–demand by users as they interact with the user interface.

3.3 Multiprocessing

In the context of Web systems, there is no *hard* multiprocessing concept such as the one needed for mission–critical monitoring applications. Instead, there are several techniques that are used to simulate multiprocessing through asynchronous data requests via AJAX or PHP. One example is a new technology known as "Web Workers" [7], which enables the execution of small pieces of JavaScript code in parallel. However, there is a limitation on the maximum number of Web Workers that can be executed simultaneously. In addition, web workers do not have access to the DOM (Document Object Model).

Java offers a very useful and powerful alternative, which is widely used in high-level languages: multithreading. With multithreading, it is possible to create several threads that can be used for different purposes. For example, developers can create one thread for each connection to legacy systems. In this way, if an application needs to connect to 20 different devices, it is possible to create 20 threads. In this scenario, each thread would execute at its own pace, asynchronously acquiring data at different frequencies. This information can be stored in shared memory areas which can be accessed by the Web browser and then presented to users. Threads can also be used to perform complex computations without affecting system responsiveness. Threads allow developers to take efficient advantage of hardware resources, especially with new multi–core processors, which allow executing each thread on a different core.

4 Developing a Mission-critical System with our Hybrid Architecture

Our hybrid architecture is not just a conceptual design. Instead, it is intended as a specification of various layers of software that enable developers to design and develop mission–critical Web–based systems in an easy and reliable manner. Therefore, developers do not need to start from scratch or dig deep in technical aspects to achieve their objective. In addition, our architecture is flexible because all modules that make up the Business, Data Exchange and Presentation layers are fully customizable.

4.1 Bussiness Layer

Our architecture's business layer is in charge of the logic to acquire and process the information that is displayed to users. This layer is comprised of data acquisition, data processing and information storage modules. This information is used by the upper layers. In this layer, events are generated in order to exchange information with other layers. It consists of a software core with structured Java objects and classes that can execute in the browser's internal memory. It also implements interoperability between Java and JavaScript by enabling access to objects in memory and making it possible to invoke Java methods from JavaScript and vice versa. In this layer, processes operate asynchronously to handle connections, data acquisition and data processing. This prevents applications from becoming unresponsive or blocked when data are being acquired or processed. With this architecture, data are available in memory and can be retrieved immediately thus, achieving excellent performance.

Figure 3 shows a block diagram that illustrates the operation and structure of this layer.

4.2 Data Exchange Layer

The Data Exchange Layer is used as a link between the objects stored in the Business Layer and the Presentation Layer. Its main task is to serve as an intermediary by using basic exchange objects, which are based on JSON (JavaScript Object Notation) in order to achieve a high degree of compatibility and reliability in data transport.

This layer also processes requests triggered by events in the Business Layer or in the Presentation Layer through data serialization. Data are encapsulated in text strings that are exchanged between objects. Next, this information is deserialized and instantiated as an object that can be used in the Presentation Layer. Since these objects are set up using an

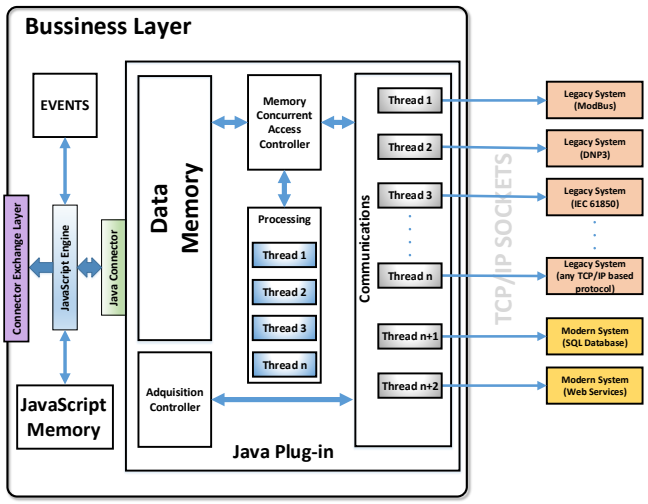

Figure 3: Bussiness Layer

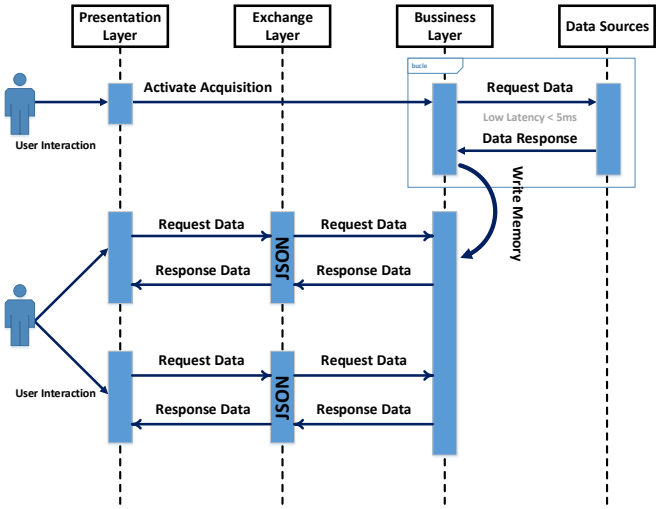

Figure 4: Sequence Diagram

open standard, they are entirely customizable and can be adapted to developers' needs.

4.3 Presentation Layer

Users interact directly with the Presentation Layer. This layer allows to set up a series of visual cues that enable users to visualize their information effectively and easily. HTML5 is used to develop the basic structure of Web pages. CSS3 is used to specify the layout of elements and to incorporate animation and interactive effects. SVG is used to create line diagrams that represent different processes. For example, in a power plant, SVG could be used to model the state of elements such as turbines, pipes, pressure indicators, valves, switches, etc. JavaScript is used to add interactive behaviors to display data in real time and to modify elements on the fly, depending on users' actions. It also possible to to dynamically create elements and insert them in HTML5 documents at run time. Hence, this layer is built with HTML5 documents that contain static information that can be modified dynamically using JavaScript.

Figure 4 shows a sequence diagram that illustrates the relation between this layer and the Exchange and Business Layers.

5 Security Considerations

While Web applications offer users attractive user interfaces and ease of use, this kind of applications present certain risks that need to be addressed [8].

Regarding security, we have implemented several techniques to enforce security measures. Since applications based on our archirecture are client–based, all operations can be performed within a closed and protected Intranet environment. In addition, the Business layer requires a certificate signed by a trusted certificate authority to operate.

Another relevant aspect is that our architecture enforces an adquate separation between browser processes and Java processes. With this approach, our software creates a secure sandbox environment for its processes.

In addition, with Java it is possible to implement diverse authentication mechanisms and data encryption mechanisms that ensure secure data transmission.

6 Conclusion

While desktop systems require users to use specific hardware, modern Web applications can be used in many other platforms including mobile phones and tablets. Mobile devices allow users to view the information they need quickly and make important decisions, even if they are out of their office. In the context of industrial monitoring systems, this has tremendous potential to improve users' workflow. For example, in power plants, operators could be viewing the information they need on site instead of being confined to an office. However, traditional Web technologies have certain limitations that makes them unsuitable for real–time, critical systems.

In this paper we have presented a hybrid architecture for developing powerful, real–time Web client applications. With our architecture, we have been able to solve two typical limitations of Web systems: lack of communication and multiprocessing capabilities. Our architecture enables developers to create modern yet reliable mission–critical systems that can be used to

monitor industrial processes.

One of the most important benefits of our architecture is that applications developed with it are platform–independent thus greatly reducing software licensing costs. In addition, our architecture is highly flexible and can be used to develop powerful Web applications for monitoring industrial applications.

7 Results

Our architecture has been used to develop a mission–critical, real–time Web system for a nuclear power plant [9]. This system has been thoroughly tested, with good results.It acquires data at very high speeds from several data souces including RTP acquisition systems, legacy systems and other devices. It performs complex computations and displays historical data, with excellent performance. Since it was developed using our architecture, it is platform–independent and can be run on any browser, in a variety of devices.

References:

[1] ModBus Organization. http://www.modbus.org/

[2] DNP Users Group, Overview of the DNP3 Protocol. http://www.dnp.org/Pages/AboutDefault.aspx

[3] OPC Foundation, The Interoperability Standard for Industrial Automation https://opcfoundation.org/

[4] Ralph Mackiewicz, "Technical Overview and Benefits of the IEC 61850 Standard for Substation Automation", SISCO, Inc. Sterling Heights, MI.

[5] Gabriella Carrozza, Roberto Pietrantuono, Stefano Russo, "Defect Analysis in Mission-critical Software Systems: a Detailed Investigation", JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS; 00:128.

[6] Jin-tae Park1, Hyun-seo Hwang1, Jun-soo Yun1 and Il-young Moon1 *Study of HTML5 WebSocket for a Multimedia Communication* International Journal of Multimedia and Ubiquitous Engineering Vol.9, No.7 (2014), pp.61-72

[7] Web Hypertext Application Technology Working Group. Web workers draft recommendation, 2010. http://www.whatwg.org/specs/web-workers/current-work/.

[8] SANS Institute. A Security Checklist for Web Application Design, 2004.

[9] Coronel Flores Efren Ruben, and Ilse Leal Aulenbacher. "A Real-Time Web-based Graphic Display System using Java LiveConnect Technology for the Laguna Verde Nuclear Power Plant". WSEAS, Advances in Information Science and Applications - Volume I, July 2014, ISBN: 978-1-61804-236-1

[10] Stankovic, John A., and Krithi Ramamritham. "What is predictability for real-time systems?." Real-Time Systems 2.4 (1990): 247-254.