DEVELOPING A COMMUNICATIONS ARCHITECTURE BASED ON WCF FOR USE IN NUCLEAR POWER PLANT SIMULATORS

Manuel Díaz, Daniel Garrido, Javier Troya
University of Málaga
Department of Languages and Computing Science
CP.29071 Málaga, Spain
{mdr, dgarrido, javiertc}@lcc.uma.es

ABSTRACT

Communications play the main role in the development of system architectures where their different parts have to continually exchange data. Windows Communication Foundation (WCF) has been designed to offer a manageable approach to distributed computing, broad interoperability and direct support for service orientation. It allows the communication among systems from any platform across intranets, extranets or the Internet, supporting at the same time a safety and reliable service. This paper presents the use of WCF in the context of distributed nuclear power plant simulators. In these simulators, communication plays a main role since they are intrinsically distributed systems. We have defined a communication architecture for the simulators using WCF for the data exchange between the different applications that compose the simulator. We also present an application developed with Visual Studio Tools for Office (VSTO). This application uses our architecture, developed with WCF, to communicate with a simulator. It has the appearance and behaviour of an Excel sheet together with some new added features and it allows us to test the communication architecture.

KEYWORDS

WCF, communications architecture, VSTO, service orientation.

1. INTRODUCTION

Before the arrival of cross-platform technologies, nuclear power plant simulators were based on old techniques and languages which hindered their scalability and reusability because of the dependence and coupling to the platform, language, operative system and other factors. The application of software engineering methodologies has allowed these simulators to be enhanced with concepts or technologies such as object-oriented programming, software components and improved middleware.

In a previous work (Díaz, M. and Garrido, D. 2004), a new generation of nuclear power plant simulators was presented, with communications based on the Common Object Request Broker Architecture (CORBA). By using CORBA and software components, a reusable and scalable communications architecture was defined, where new components could be added to existing simulators. With the arrival of new technologies such as Web Services (W3C Working Group, 2007), more advantages and future scalability could be achieved. A Web Service is a collection of protocols and standards used to exchange data between applications.

Using these Web Services (WS), a new architecture was developed to communicate end user applications with the simulator (Cortés, J.A. et al., 2008). The objective of these applications is to train future nuclear power plant operators in a variety of operation and maintenance procedures, allowing them to rehearse different situations, from the most usual ones, such as temperature monitoring, valve operation, etc. to the most unusual ones like handling emergency situations.

On this occasion, we have made use of Windows Communication Foundation (Windows Communication Foundation, 2008), a more innovative technology than Web Services. WCF is a runtime and a set of APIs for creating systems that send messages between services and clients. The same infrastructure

and APIs are used to create applications that communicate with other applications on the same computer system or on a system that resides in another location and can be accessed via the Internet. Using this technology, a new communications architecture has been developed. Its objective is to communicate the simulator with different end user applications developed with different programming languages such as C#, Visual Basic or Java. So far, the use of WCF may seem similar to Web Services. But, unlike them, WCF offers binary encoding, which is useful for efficient and rapid data transfer, whereas WS only offer the possibility of text encoding. "Fast Web Services" are an option to enable binary exchange in WS (Sandoz P. et al., 2003).

We have also used Visual Studio Tools for Office (VSTO), which helps to extend Microsoft Office applications using Visual C# and Visual Basic. With this technology, we have developed an end user application which inherits the Excel behaviour and interface. Its main intention is to check the behaviour of simulator models, so it is very useful in the development of simulators.

The structure of the paper is as follows. Section 2 presents the system architecture developed. Section 3 describes the new technology used, WCF. Section 4 explains the use of WCF in our architecture and compares it with WS. In section 5 the Excel client application is presented and explained. Finally, the conclusions extracted from this whole study and the benefits of this new integration are presented.

2. SYSTEM ARCHITECTURE

As stated previously, we have made use of WCF, which is a fundamental part of the whole structure, to develop our architecture (see Figure 1).

In this architecture, the core of the simulator is the application which deals with the execution of the simulation models. It is mainly developed in C/C++ and it contains all the logic of the simulator. The WCF service sends requests, which are received from its client applications, to the core of the simulator. Then, the simulator provides the service with the data requested and the service sends the data received to the clients. Requests may be of different types (see section 4) depending on what clients want.

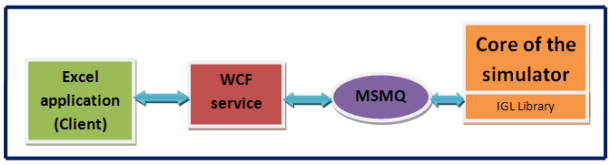

Figure 1. System architecture

The WCF service is developed in C#, whereas, as mentioned before, the core of the simulator is developed in C/C++. For these two parts to communicate, Microsoft Message Queuing (MSMQ) has been used (Message Queue Server, 2007). This technology allows applications developed in different programming languages and running at different times to communicate across heterogeneous networks and systems that may be temporarily offline. The applications send messages to the queues and they are read later from these queues by other applications (see section 4). The WCF service does not communicate directly with the core of the simulator, but through the IGL library, developed in C++. This library reads the messages that the service sends to the queues. Then, it informs the simulator of the request received. Equally, the data flows the same way in the other direction; the simulator transmits the result of the request to the IGL library and this sends messages to queues which are then read by the WCF service.

We have used this library because the core of the simulator does not have to deal with the sending and receipt of messages to/from the queues. In this way, our architecture does not depend on the core of the simulator. Consequently, interoperability is increased between the WCF service and the IGL library and more extensibility is provided, because if we need to expand communications between the service and the core in the future, we will have to modify only the former (i.e., the service) and not the latter (the core), which would be a very complex and time consuming task.

The Excel application is the one which uses this communications architecture. It communicates with the WCF service. Depending on what it needs, it makes requests to the service and receives data from it. The Excel application has been developed with VSTO (see section 5) and the C# language.

3. WCF TECHNOLOGY

Nowadays, the development of distributed applications running on different machines connected by networks is increasingly common. Some technologies normally used to make this communication possible are Java RMI (Grosso W., 2001) or CORBA (Henning M. and Vinoski S., 2001) which are reliable and scalable to some extent in an intranet environment but not over the Internet.

A well-known technology which deals with communications among different platforms and across networks is Web Services. However, we have preferred to use Windows Communication Foundation. It has been specifically designed to offer a manageable approach to distributed computing, broad interoperability, and direct support for service orientation.

WCF has several important advantages over Web services. It supports more protocols for transporting messages than WS, which only support sending messages using HTTP. WCF supports sending messages using HTTP, as well as TCP, named pipes, and MSMQ.

WCF is based on the notion of message-based communication. Messages are sent between endpoints. Endpoints are places where messages are sent or received (or both), and they define all the information required for the message exchange. A service exposes one or more application endpoints and the client generates an endpoint that is compatible with one of the service's endpoints. An endpoint describes in a standard-based way where messages should be sent, how they should be sent, and what the messages should look like.

Once we decided to use WCF for the system architecture, a problem arose when trying to obtain data from the core of the simulator since it is written in C/C++, while WCF uses C#. After searching for a way to communicate them, the best solution found was Microsoft Message Queuing (MSMQ). It enables applications running at different times to communicate across heterogeneous networks and systems that may be temporarily offline. Applications send messages to and read messages from queues.

4. USE OF WCF

According to the architecture of this system, when the WCF service receives requests from clients, it communicates with the core of the simulator by means of MSMQ to provide the clients with what they want.

Clients communicate with the simulator by calling the methods of the WCF service in a transparent way. The WCF service is made up of methods for dealing with the management of client applications connected to the WCF service, methods to change or obtain the state of the simulator (whether it is running or not, the mode and the speed of simulation…), methods for variable monitoring and methods to detect the errors thrown by the simulator.

All these methods communicate with the IGL library by sending and receiving messages to/from the MSMQ queues. In this way, the WCF service tells the IGL library, by means of MSMQ messages, what it wants the simulator to do. Then, if the WCF service expects any response, the IGL library sends a message with the response given by the simulator.

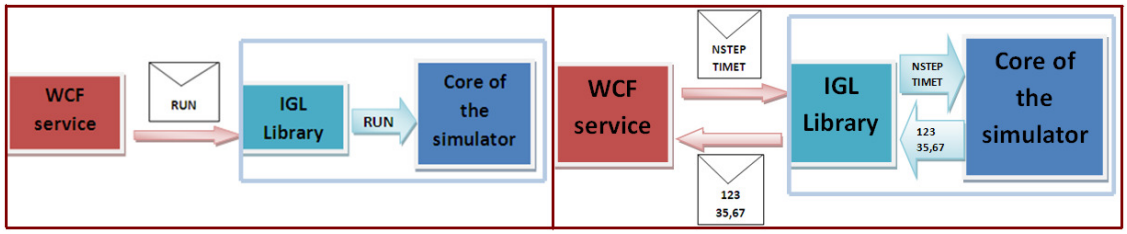

Figure 2. Communication between the WCF service and the core of the simulator

In figure 2 we can see two examples of communication between the WCF service and the core of the simulator. In the first one, the service is telling the library that the simulator must start running. Here, the simulator does not need to send any data back to the service. In the second example, the WCF service requests the value of two variables. Now, the simulator sends back the value of both variables.

4.1 Performance comparison between WS and WCF

To test the performance of the proposed architecture, tests have been developed using a web service instead of our WCF service. The web service executes the same actions as our WCF service does. A comparison has been made of the time both technologies consume when providing variable values to the Excel client application. That is to say, we have recorded the time that elapses between when the client asks the service for variable values to when the service provides the client with these values.

There are two technologies that influence the time consumed; the service technology (WS or WCF) and MSMQ. Despite the time that MSMQ consumes in both architectures, which is marked, there is a big time difference when using WS and WCF. Figure 3 shows a diagram where the two architectures are compared. The difference in velocity is significant. When the client requests the value of 10 variables, the values are received in 15 ms with WCF and in 156 ms with WS. When the variables requested are 100, the values are received in 234 ms with WCF and in 1953 ms with WS.

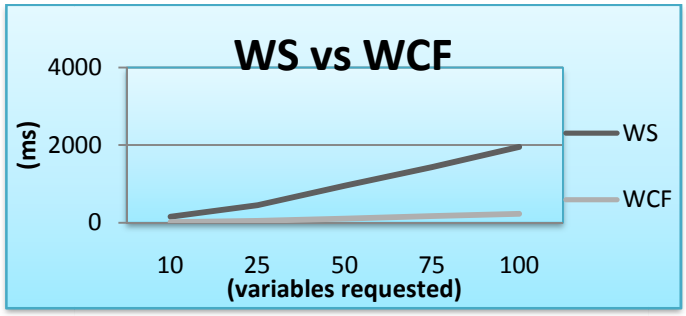

Figure 3. Web Services vs Windows Communication Foundation

In the configuration of our WCF service, the transport protocol was set to TCP, and the encoding was binary. The only transport protocol available for WS however is HTTP and the encoding possible is text. This is the reason why this WCF configuration is much faster than WS. WCF communication over the TCP transport protocol is really more efficient than over HTTP and since the data transmitted is binary, there is no need to either encode or decode it.

5. CLIENT PART: EXCEL APPLICATION (VSTO)

This application uses the architecture developed to communicate with the simulator. It sends orders and requests in a transparent way by means of the WCF service to the core of the simulator and waits for data from it.

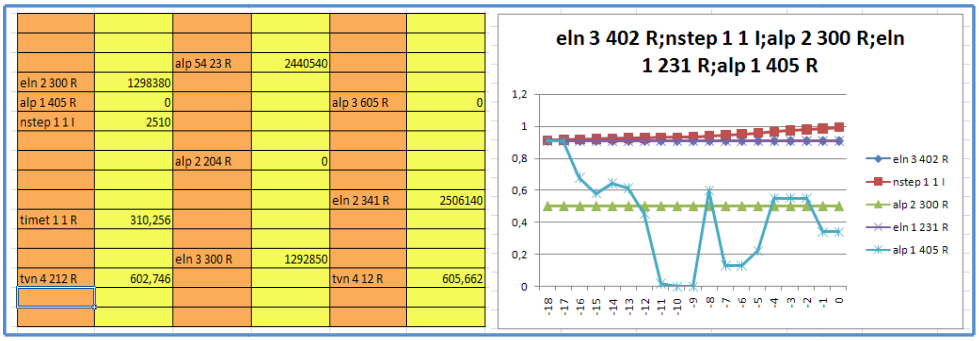

Figure 4. Monitoring variables and displaying a chart

The use of VSTO has been very useful for the development of this application. It inherits all the behaviours and the interface of Excel applications. This Excel application has been developed to make the

simulator run and stop, change the velocity of the simulator and the mode of simulation, obtain continually the state of the simulator, display the errors of the simulator and monitor their variables, change the value of these variables and display charts which show the value of these variables. The application presents the Excel interface with a control panel added. With this control panel, the user can send orders and requests to the simulator by clicking a button or changing the value of a text field. This control panel also displays the state of the simulator, by showing some parameters.

The cells of this Excel application have an additional functionality since they can monitor the value of the simulator variables. The user can write the names of variables in the cells on the left (darker coloured cells) and their values are monitored in the cell on the right (lighter coloured cells) (see figure 4). Moreover, users can select, by means of the control panel, which variables they want to see in a chart (figure 4).

6. CONCLUSION

Many companies, while trying to adapt to the evolving technology, study how to take advantage of the new research and technologies in terms of the time cost benefits.

This paper has presented the development of a communications architecture in the field of nuclear power plant simulators. The architecture uses Windows Communication Foundation, the most innovative communication technology which has several important advantages over Web Services. Using this technology, the development of the communication infrastructure is easier than with traditional techniques such as RPC or sockets.

This structure has significant advantages. It is easily extensible, so that functional qualities can be added without difficulty in future versions. It is a distributed system, the WCF service and the clients may be in the same or in different machines, they can communicate via the Internet. In addition, the developed architecture is scalable and, therefore, new end user applications developed in the future can communicate with the simulator by means of the WCF service, so that they would act as new clients in the architecture.

We have also made use of Microsoft Message Queuing, which allows us to communicate two parts of this architecture written in different programming languages, C++ and C#. Both parts send messages with the data they want to exchange. These messages are sent to some queues and are then recovered by the other part, which interprets them.

To prove the whole architecture, a client application has been developed using Visual Studio Tools for Office. It inherits the behaviour of an Excel application with some other features added. These features are related to the communication of the application with the simulator by means of the architecture developed. This way, the application can monitor different simulator variables, it can also switch on/off the simulator, change the simulation speed/mode, display the errors generated in the simulator…

REFERENCES

Cortés, J.A. et al., 2008. A New Generation of Nuclear Power Plant Simulation Applications. *The 22nd annual European Simulaion and Modelling Conference, ESM*.

Díaz, M. and Garrido, D., 2004. A Simulation Environment for Nuclear Power Plants. *8th IEEE International Symposium on Distributed Simulation and Real Time Applications*, pp. 98-105.

Grosso W., 2001. *Java RMI*. O'Reilly Media.

Henning M. and Vinoski S., 2001. *Advanced Corba Programming with C++*. Addison Wesley.

Message Queue Server, 2007. *Microsoft Technet*. Online document at http://msdn2.microsoft.com/en=us/library/ms711472.aspx.

Sandoz P. et al., 2003. *Fast Web Services*. Online article at http://java.sun.com/developer/technicalArticles/WebServices/fastWS/.

W3C Working Group, 2007. *Web Services Architecture*. Online documents at http://www.w3.org/TR/ws-arch/.

Windows Communication Foundation, 2008. *Microsoft Technet.* Online documentation at http://msdn.microsoft.com/en-us/netframework/aa663324.aspx.