Developing a Web-Enabled HLA Federate Based on poRTIco RTI

Zhiying Tu, Gregory Zacharewicz, David Chen

Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

DEVELOPING A WEB-ENABLED HLA FEDERATE BASED ON PORTICO RTI

Zhiying Tu
Gregory Zacharewicz
David Chen

University Bordeaux
IMS/LAPS, 351 Cours de la Libération
33405 Talence, FRANCE

ABSTRACT

This paper presents an approach to implement distributed simulation software to test, validate and improve information exchange between enterprises. The approach assumes some of the recently released HLA Evolved standard web service requirements. The implementation is based on improving the open source poRTIco HLA RTI tool. The paper mainly focuses on the presentation of the HLA web-enabled federate proposed to fulfill the web service needs of the brand new HLA 1516-2010 standard. The idea is that a federate is being connected to an HLA federation based on previous HLA standards version (1516-2000 or 1.3) and, in parallel, to other federates outside the LAN federation via the WAN. This approach intends to improve federate components interoperability and agility in a heterogeneous, distributed and dynamic context. To achieve that goal, time management mechanisms have been developed. The concept is validated through distributed simulation of a car selling enterprise modeling test case.

1 INTRODUCTION

Enterprises exchange information in an increasingly dynamic context where distribution and heterogeneity are key barriers to be crossed. On the one hand, one way to facilitate crossing these barriers is to model and simulate the system first. On the other hand, the actual challenge for data exchange resides in their capacity to be available on demand thanks to services proposed on the web. This paper contributes to distributed simulation HLA Evolved web service development that meets those two requirements. The approach adopts the federate development life cycle based on the "harmonized and reversible development framework for HLA application" proposed in Tu, Zacharewicz, and Chen (2010). This cycle facilitates to bridge the gap from the enterprise concepts to implementation and offers a new reversible approach. The paper's main focus is to create a "web service" federate component to increase the HLA distributed simulation interoperability. It presents an implementation of Web Service federates based on the open source software poRTIco RTI (poRTIco 2009).

After providing background, the first part of this paper presents the architecture of this framework. Then it details the algorithm to communicate and to synchronize the components of this framework. Finally, an example with web service federates models of car selling enterprises is introduced to demonstrate feasibility of achieving better interoperability and agility for data exchange of enterprise information systems in a web dynamic context.

2 BACKGROUND

2.1 High Level Architecture

The High Level Architecture (HLA) is a software architecture specification that defines how to create a global software execution composed of distributed simulations and software applications. This standard was originally introduced by the Defense Modeling and Simulation Office (DMSO) of the US Department of Defense (DOD). The original goal was reuse and interoperability of military applications, simulations and sensors.

In HLA, every participating application is called federate. A federate interacts with other federates within a HLA federation, which is in fact a group of federates. The HLA set of definitions brought about the creation of the standard 1.3 in 1996, which evolved to HLA 1516 in 2000 (IEEE 2000).

The interface specification of HLA describes how to communicate within the federation through the implementation of HLA specification: the Run Time Infrastructure (RTI). Federates interact using services proposed by the RTI. They can notably "Publish" to inform about an intention to send information to the federation and "Subscribe" to receive some information created and updated by other federates. The information exchanged in HLA is represented in the form of classical object oriented programming. The two kinds of object exchanged in HLA are Object Class and Interaction Class. Object class contains object-oriented data shared in the federation that persists during the run time; Interaction class data are just sent and received information between federates. These objects are implemented within XML format. More details on RTI services and information distributed in HLA are presented in IEEE (2000).

The FEDEP (Federation Development and Execution Process) describes a high-level life cycle for the development and execution of HLA federation. FEDEP proposes a seven-step process to guide the development of the simulation system through phases of requirements, conceptual modeling, design, software development, integration, and execution (IEEE 2003). It has been updated and renamed as DSEEP (Distributed Simulation Engineering and Execution Process) in IEEE (2010).

2.2 HLA 1516-2010 Evolved

HLA Evolved is the nickname for HLA IEEE 1516-2010 standard, which is built upon the previous HLA 1516-2000 and HLA 1.3 standards (Möller et al. 2008). The HLA 1516-2010 standard has been published recently in August 2010. HLA Evolved gives developers and users of distributed simulation a large number of new development, deployment and net centric capabilities. HLA Evolved contains the functionality of HLA 1516-2000 but it also adds a number of new features, including:

- Fault tolerance support services
- Modular FOMs
- Web Services (WSDL) support/API
- Smart Update rate reduction
- Encoding helpers
- Extended XML support for FOM/SOM, such as Schemas and extensibility

2.3 Model Driven Architecture (MDA)

The Model Driven Architecture (MDA) methodology is defined and adopted by the Object Management Group (OMG 2003) is designed to promote the use of models and their transformations to consider and implement different systems. It is based on an architecture defining four levels, which goes from general considerations to specific ones.

- CIM Level (Computation Independent Model): describes the entire system and its environment
- PIM Level (Platform Independent Model): models the sub-set of the system to be implemented

- PSM Level (Platform Specific Model): takes into account the specificities related to the development platform
- Coding Level: consists in coding the software or more generally the enterprise applications

2.4 Harmonized and Reversible Development Framework for HLA Based Interoperable Application

With the goal of boosting HLA federation development using existing (legacy) information systems, a bi-directional development lifecycle has been proposed in Tu et al. (2010), on top of FEDEP (now DSEEP) and MDA. Many approaches have been proposed to merge those two development lifecycle (e.g., Tolk and Muguira 2004). The approach described in Figure 1 is entitled "Harmonized and Reversible Development Framework for HLA based Interoperable application." It differs from previous approaches by its two development directions (developing and reverse engineering). Its strong features indicated by the key words used are described below.

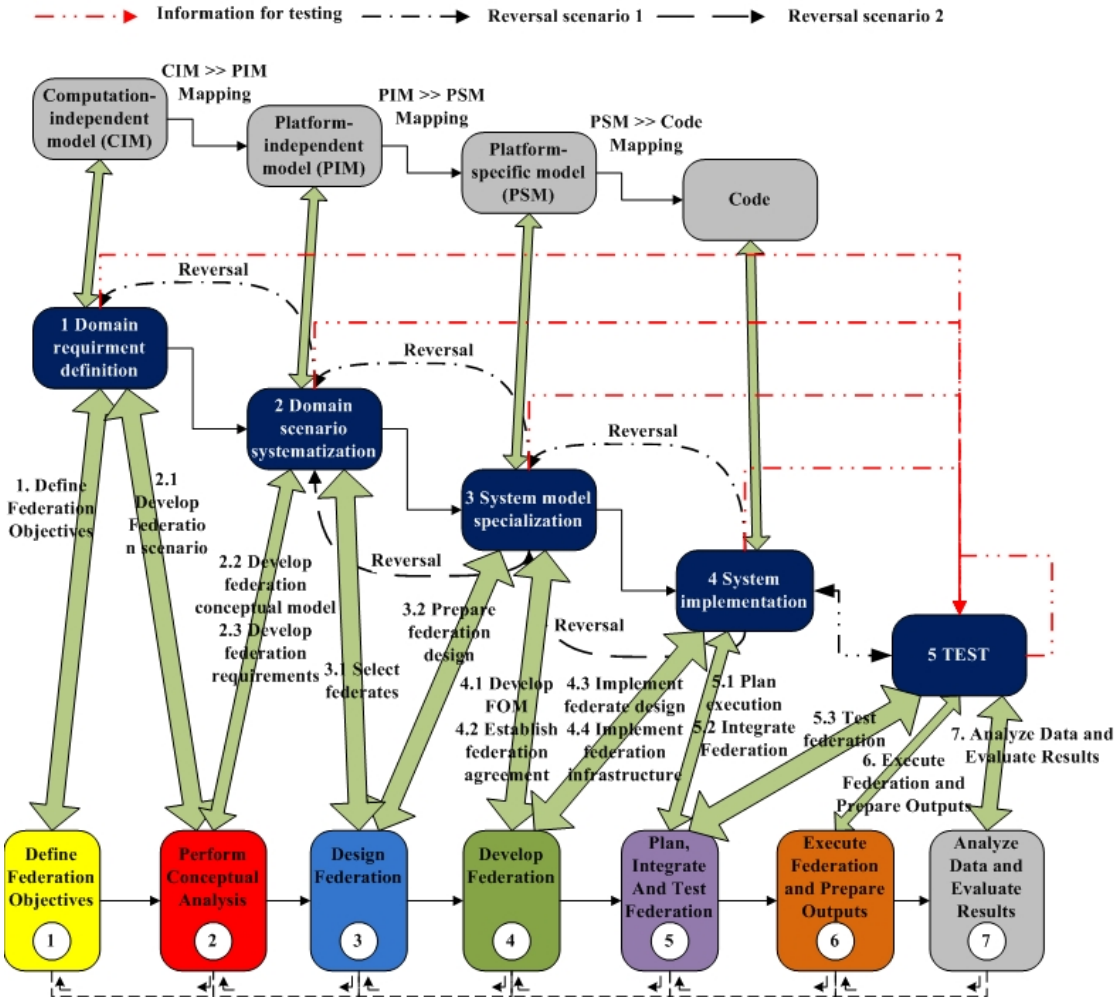

Figure 1: Harmonized and reversible development framework

Harmonized means that the framework consists of several techniques. As Figure1 shows, in this framework, there is a new five step development life cycle which aligns MDA and HLA FEDEP and a reversible modeling process. In addition, the framework also uses web services to improve the performance of HLA.

Reversible means that the framework uses model reverse engineering technique to discover the model from the legacy system. The model reverse engineering technique is used to avoid rebuilding the legacy system for different cooperation. Our objective is to accelerate the development and reduce the cost.

HLA means that the framework is dedicated to the development of HLA based application. And, poRTIco RTI is chosen for our research. The reason is not only because it is free, but also we want to initiate an open framework to receive more comments and contributions from people who are interested in our research.

Interoperable means that the framework provides a solution for achieving enterprise interoperability. It will be used to overcome the enterprise interoperability barriers and to help realize the short-lived ontology approach, which is mentioned by Zacharewicz, Chen, and Vallespir (2008, 2009).

2.5 poRTIco RTI

PoRTIco is a fully supported, open source, cross-platform HLA RTI implementation. The PoRTIco project was initially founded by Tim Pokorny and Michael Fraser in 2005 and was known as jaRTI. Then, after a considerable amount of internal development, the first public release was made during SimTecT in June of 2006. Following its initial debut, jaRTI development continued to advance, with additional features, services and tools being added over the next year. The project also received a lot of favorable attention and up to the middle of 2007 it had been downloaded more than 1300 times. In May 2007, the PoRTIco Project became the subject of funding received from the Australian Defense Simulation Office (ADSO) (PoRTIco 2009).

PoRTIco RTI is designed with modularity and flexibility in mind. It is intended to provide a production grade RTI implementation and an environment that can support continued research and development. It is licensed under the terms of the Common Developer and Distribution License (CDDL) and is actively developed and maintained by its team of core contributors.

3 SPECIFICATION OF WEB-ENABLED HLA FEDERATE DESIGN

3.1 Overview of HLA Evolved WS Scenario

As known, HLA has been developed within the defense simulation community with a focus on simulation in Local Area Network or in Virtual Private Network. Its disadvantages include: lack of flexibility, lack of integration with business, etc. Meanwhile, Web Services concept has been developed within the commercial enterprise community, and has achieved a great success in business domain. Its success stems from the good characteristics of the technology itself, its wide recognition by enterprises and business organizations, and effective support of the open source community (Richardson and Ruby 2007). As a result, Web Services is the perfect option to help HLA overcome its disadvantages.

Our general idea of HLA evolved Web service is illustrated in Figure 2. We assume that a cooperative project has been launched between several partner enterprises. The information systems of the members run well within the federation. During this project, other enterprises want to join this project with different expectation, such as different cooperation time periods, different cooperation domains, different expected results from federation, etc. As a result, to rebuild the federation is impossible. Our solution is to add one particular federate, *WebservicesFederate*, as shown in Figure 2. This federate will provide various services, different access permissions, and a common API for connecting the federation. The candidates can use the common API and the service they prefer to generate their own web federate, which will be used to connect to the existing federation with different authorities via the wide area network.

For example, in Figure 2, we assume that two enterprises X and Y want to participate in an existing project. Enterprise X is a supplier and enterprise Y is a client who is interested in the final product of this project. Thus, enterprise X has to know the workflow which relates to his business, and synchronize its information with other stakeholders. Enterprise Y only asks for information from the federation, so, it doesn't have to synchronize with other systems. As a result, enterprise X will ask *WebservicesFederate*

for the service with an authority of synchronization with other federates, while enterprise Y will only require the service with the lowest authority, that allows listening to the federation. Finally, no matter what kinds of services they get, they are connected to the existing federation via web services.

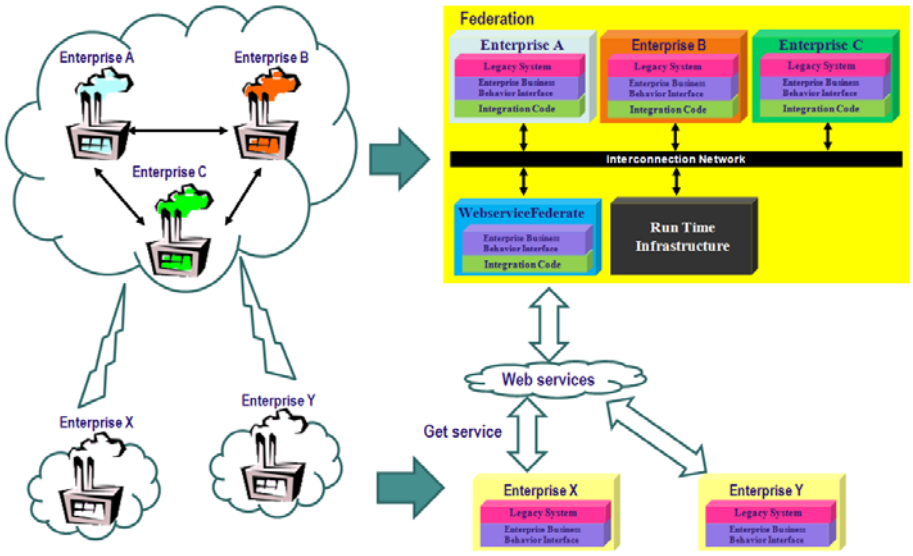

Figure 2: HLA Evolved Web Services

3.2 Architecture of HLA Evolved WS

The architecture of HLA Evolved Web Services is illustrated in Figure 3. In our project, we chose the open source RTI, poRTIco, which doesn't provide Web-RTI functionality. Actually, at the beginning of this effort a RTI tool named pRTI Evolved was announced to fulfill 1516-2010 and to support Web-RTI functionality (Möller and Löf 2005; Möller, Clarence, and Mikael 2007). However time management was still under development in this tool and it remains in commercial domain. This reason and our intention to develop an open framework has guided the choice to add functionalities to poRTIco. Thus, we implement *WebservicesFederate* as a bridge to take charge of providing web services, and connecting and synchronizing federates outside the federation with federates inside the federation. As mentioned in Tu, Zacharewicz, and Chen (2010), after the harmonization of MDA and HLA FEDEP, we have integrated code which provides a RTI tool independent API. This API can be reused and published as common API. So, the candidates can follow the model reversal scenario described in Figure 1, to generate their own Enterprise Business Behavior Interface based on the common API. After that, the new federate outside the federation can send the information to bridge via the Web services interface and be synchronized by bridge.

3.3 Web Services Federate Design

A schema of web services federate design is illustrated in Figure 4. In this design, web services federate is a special federate, which is inside the Local area network (LAN) but not fully included in federation, and a part is connected to the WAN. According to this specificity, web service federate is divided into two parts:

- *WebServicesBridge*, which is inside the federation;
- *WebServicesServer*, which is straddling between outside the federation connected to the WAN, and being inside the LAN.

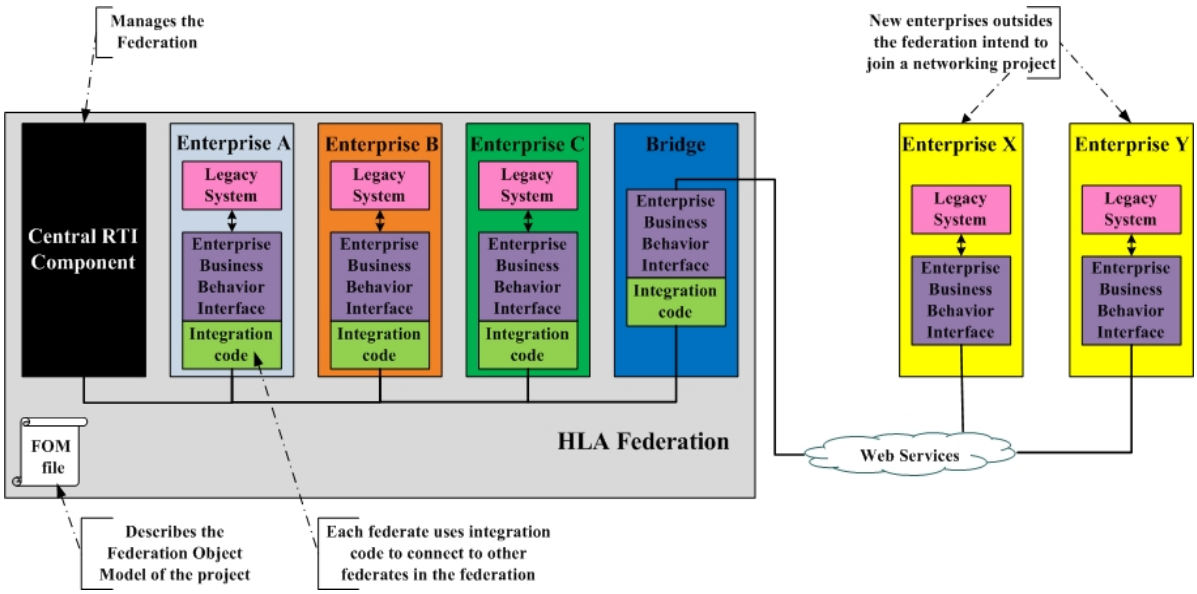

Figure 3: Architecture of HLA Evolved Web Services

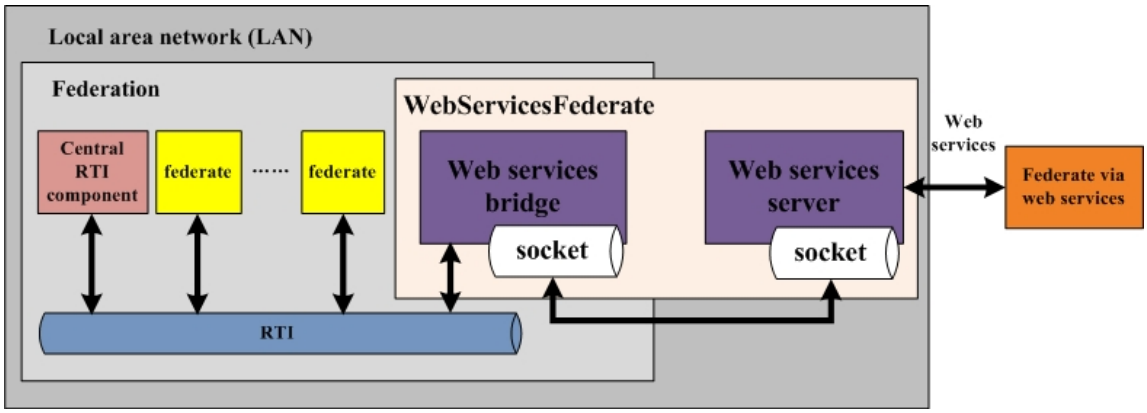

Figure 4: Web services federate design

These two parts are connected by socket. This design customizes poRTIco RTI that was not originally designed for Web RTI functionality. In order to implement Web RTI functionality, *WebServicesBridge* and *WebServicesServer* are designed for web service federate.

- *WebServicesServer*: is used to publish web services interface to potential customers outside the federation. It takes charge of monitoring and replying to the federate via web service. When this server receives the message from the federate outside federation, it will generate a User Datagram Protocol (UDP) data package and send it to *WebServicesBridge* by socket connection.
- *WebServicesBridge*: is used to synchronize the message from the federate outside federation with other federates inside the federation. This bridge transmits messages to federates inside federation by RTI, but exchanges messages with web services server by socket connection. When web services federation establishes, this bridge will launch a thread to monitor the events happening in *WebServiceServer*.

3.4 General Solution for Failure Tolerance

As we involve Web services and UDP in our methodology, the failure tolerance needs to be considered. In this paper, we consider two failures: data exchange delay and data package lost.

First, let's describe this simulation and define the major elements in this simulation. Because our simulation is time-step real-time simulation, the time-step (simulation time unit) needs to be defined first. Thus, as shown in Figure 5, we assume that the simulation time unit (Δt) of the federation is 3 seconds, which means every 3 seconds will trigger a new event. We use the conservative algorithm described in Fujimoto (2000) and Zacharewicz, Frydman, and Giambiasi (2008). For example, in Figure 5, Federate A sends one event with Time stamp (T_{stamp}) plus L_A (Lookahead of A) equals 3 to the event queue, so when simulation time passes one Δt, this event will be triggered. Every federate can announce its event with T_{stamp} plus Lookahead. Lookahead is a special non-negative value, which establishes the lowest value of time stamps that can be sent in its Time Stamp Order (TSO) messages. In our simulation, we assume that the lookaheads of WebServicesfederate and federates outside of federation are 0, meanwhile, the lookaheads of federates inside of federation are bigger than 0 and depends on their own process. When simulation time moves forward, RTI will send $Event_j$ of $federate_j$, whose $T_{stampj} + L_j > LBTS_i$ (Low Bound on Time Stamps) will be triggered and sent to related $Federate_i$. Due to the performance of Web Services and UDP and our simulation context, we also assume that each federate can store three states, S_C, S_{P1}, and S_{P2}. S_C is the current state. S_{P1} is the state one Δt time ago. S_{P2} is the state two Δt time ago. The reason of saving three states is to backup necessary information in order to answer overdue customer's request from *WebServiceFederate*. The reason of only saving three states is to limit the times of re-ACK(ACKnowledgment) between *WebServiceFederate* and federate outside LAN, which can ensure the message channel between *WebServicesbridge* and *WebSerivesserver* stays responsive, there is strict control on the increase of each federate's memory, and redundancy exists in the federate. In addition, in our simulation context, the time step allows federates inside LAN to keep their current state for a quite long period, so three backup states are enough for querying (We will not roll back the state because of overdue customer request. We only provide the state query service, so this roll back querying will not affect the message synchronization inside the federation). Normally, we also believe that if no reply is received after the third PING (Packet Internet Grope), the web connection is broken.

3.4.1 Data Exchange Delay

For example, in Figure 5, federate C sends one message with time stamp plus L_C equals 9 to *WebServicesFederate*. Normally, when *WebServicesFederate* receives this message, the current simulation time ($T_{current}$) should be less than T_{stamp} plus L_C, but, if this message transmission has several seconds time delay, when this message arrives $T_{stamp} + L_C$ might be $< T_{current}$, which means this event has already expired. As a result, there is no reply for federate C. The solution for data exchange delay is if $T_{current}$ is more than $T_{stamp} + L_i$ of $message_i$, *WebServicesFederate* will ask for the past state of request federate for simulation. However, if the authority of $message_i$ (MA_i) is low, the federation will ignore this message.

3.4.2 Package Lost

For example, in Figure 5, federate D sends one message with T_{stamp} plus L_D equals 12 to *WebServicesFederate*. However, the package is lost during the web transmission, and this message cannot join the simulation of federation before its own time stamp. As a result, there is no reply for federate D. The solution for package lost is to set the attribute in federate D called waiting time (T_{wait}). If T_{wait} is bigger than Δt, then federate D will resend the message. And the maximum resend time (F_{resend}) is twice. If *WebServicesFederate* receives the message, it will calculate the time difference ($T_{difference}$) and decide which state of the requested federate will be used to reply. Again, if the authority of the message is low, federation will ignore this message.

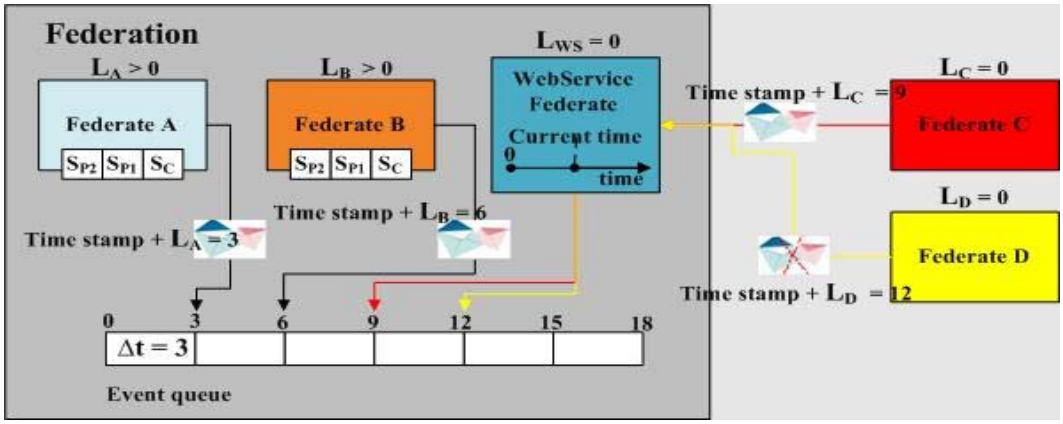

Figure 5: General solution for failure tolerance

The general algorithm of the failure tolerance is provided in Figure 6.

- For a federate outside the federation:

```
Fresend = 0;
while (Fresend < 2) {
      if ( Twait > Δt ){
            resend message;
            Fresend++;
      } else {
            Fresend = 2;
      }
}
```

- For a federate inside the federation:

```
while ( simulation time passes Δt ){
      SP2 = SP1;
      SP1 = SC;
      SC = runSimulation();
}
```

- For the *WebServicesFederate*:

```
if (Tcurrent > Tstampi + Li) {
      if (MAi != low) {
            Tdifference = (Tcurrent - Tstamp - Li)/Δt;
            switch ( Tdifference ) {
         case 0 : state = SC;
                              break;
         case 1 : state = SP1;
                  break;
         case 2 : state = SP2;
                        break;
         case 3 : ignore message;
              break;
         default : ignore message;
                              break
                        ;
            }
      }else{
            Ignore message;
      }
} else {
      if (Tstampi + Li > LBTSj){
            send event to Federate j;
            state = runSimulation();
      } else {
            state = SC;
      }
}
```

Figure 6: Failure tolerance algorithm

4 CASE STUDY OF WEB-ENABLED HLA FEDERATE DESIGN

4.1 Simulation statement

Based on the design in the previous sections, a demonstration for HLA evolved web services simulation has been proposed by a Master's student of IMS Bordeaux during his internship period. This simulation is based on poRTIco RTI and Java. It is implemented on Eclipse and is portable over MS Windows, UNIX or Mac systems with JDK 1.6.0 (or higher) environment and poRTIco environment.

This case describes a car manufacturing work-in-progress tracking system. The actors of this case are car sales agency, car manufacturing factory and suppliers of wheel and engine. The actions of this case are simply defined as purchase, manufacture and delivery. The goal of this case is to use our methodology to achieve zero inventories for car manufacturing factory and order just in time for supplier. And the most important thing is that the car sales agency can track the order from the beginning to the end in real time.

The simulation deployment is illustrated in Figure 7. We establish a traditional HLA federation in a LAN where we set up some federates as end-users interfaces. In addition, in order to enable the web services federate, we deploy one server as the web service bridge inside the federation, and deploy another server as the web service server outside the federation but still inside the LAN. And these two special servers will be connected by socket connection. To ensure the security of this connection, the socket connection ports are designated and data encryption/decryption is enforced on both sides of web service server and web service bridge. After web service server is deployed, JAX-RPC based web service interface of HLA federation is released. Then, different federates outside the federation can link to the federation via diverse web connections.

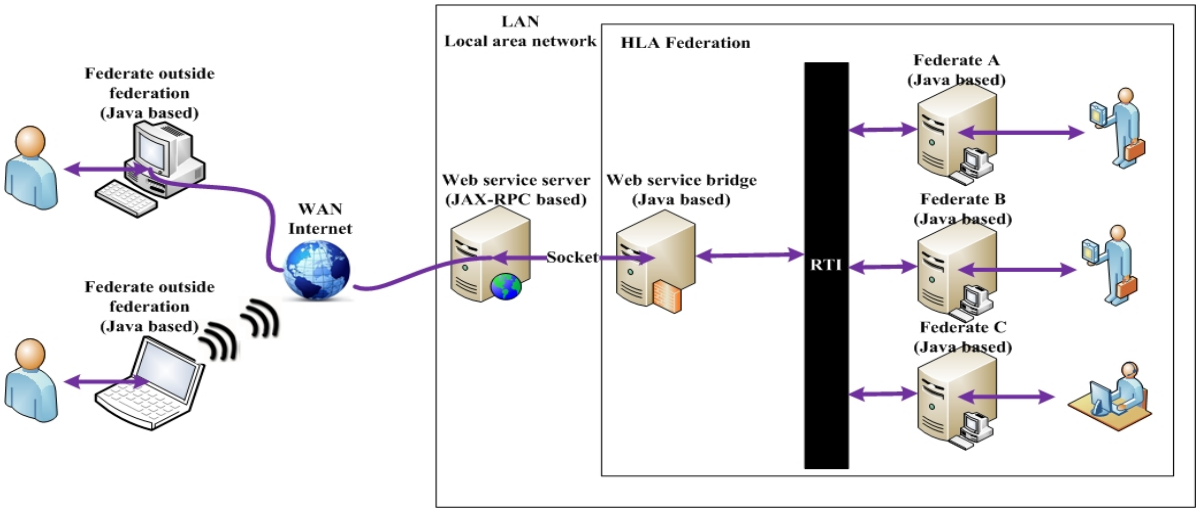

Figure 7: Simulation deployment

4.2 Case Study Simulation Specification

The use case diagram of this case simulation is shown in Figure 8. Once the Car sales agency (CSA) receives an order from a client, it sends an order to the Car manufacture factory (CMF), and then CMF will calculate the amount of raw materials based on the bill of materials. After that it dispatches orders to different suppliers, such as Wheel suppliers (WS) and Engine suppliers (ES), to get the parts or raw materials. When suppliers finish the production, they deliver the products to the CMF who will assemble them and then deliver to the CSA. In this simulation, the status of the order is the issue of concern. CSA cares about when it can receive the cars it orders, and it is concerned about the status of the associated manufacturing process.

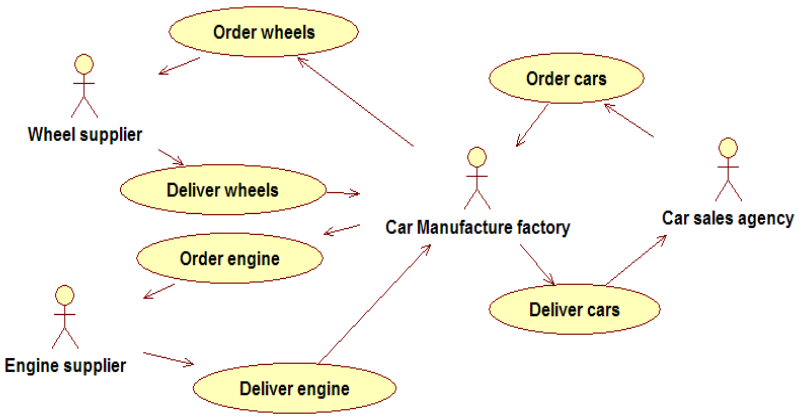

Figure 8: Use case diagram of case study

The class diagram of this demonstration is illustrated in Figure 9. In this simulation, there are various kinds of federates, and according to each federate, there is a correlated federate ambassador. Thus, factory design pattern is used to create flexible federates. While, for web service federate, besides the inheritance from abstract "federate" class, it also implements the *HLAWebServiceInterface*.

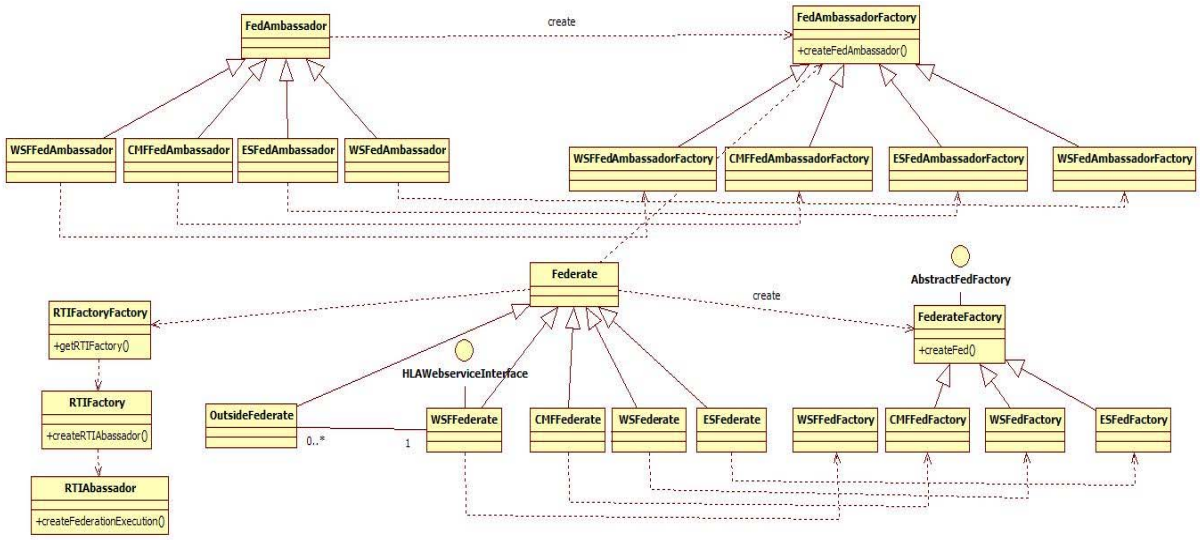

Figure 9: Class diagram of case study

The object class structure table is shown in Table 1. We define four parameters for each federate: "dayToFinish" shows the remaining days in the manufacturing process, "currentState" shows the current state of the order, "count" shows the completed product, and "Price" represents the current cost.

We assume that CSA is the federate outside the federation, while, the WS, ES and CMF are federates inside the federation. CSA is synchronized with WS, ES and CMF by the bridge. Figure 10 illustrates the sequence diagram of this simulation. CSA requires the status of the order through web services, and bridge uses socket to connect the federation, to listen to the federation, and for waiting for the answer.

Table 1: Object class structure table

HLA Object root	**WheelSupplier**	**dayToFinish**
		currentState
		count
		price
	CarManufacturer	**dayToFinish**
		currentState
		count
		price
	EngineSupplier	**dayToFinish**
		currentState
		count
		price

4.3 Case Study Simulation Implementation

The illustration of simulation user interface of Car sales agency and car manufacture factory is shown in Figure 11.

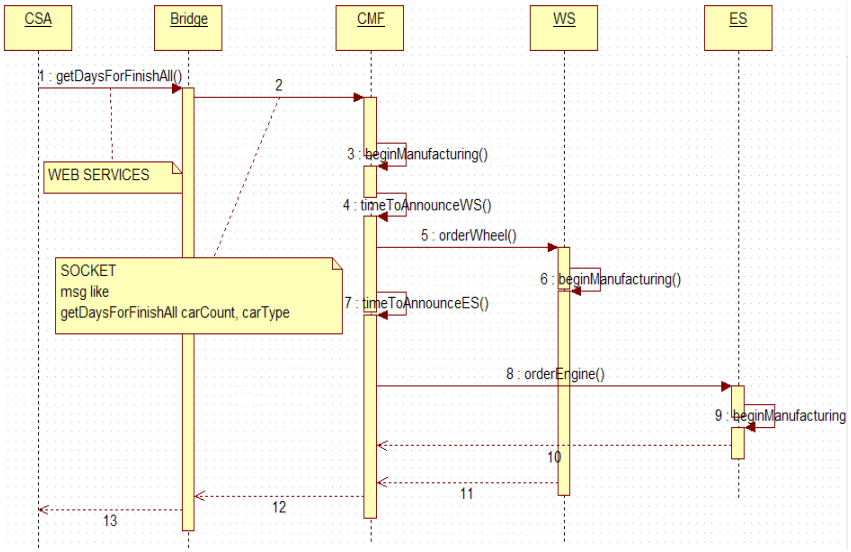

Figure 10: Sequence diagram of case study

The staff of car sale agency can input the number of cars for one order. After the confirmation of this order, the client side displays the total manufacturing days. During the car manufacturing, car sales agency can request detailed information of the manufacturing process, and the federation will immediately send back the result, which has been generated as a bar chart to vividly illustrate the status of the manufacturing process.

In car manufacturing factory central control interface, the remaining days in the car manufacturing process will be presented. In order to assist the car assembling process, the information of suppliers will be illustrated. By clicking the “refresh” button, car manufacturing factory can receive the latest information from each supplier.

5 CONCLUSION

The simulation of a HLA Evolved *WebServiceFederate* has been carried out with the open source RTI, poRTIco extended by a new component straddling between HLA federation LAN and WAN. This new component, useful to facilitate the enterprise data exchange simulation, attempts to fulfill one of the new features specified in HLA 1516-2010. The simulation runs properly with laboratory data, after filling in some gaps between HLA Evolved approach and the API provided by poRTIco. We point out that our solution is in the application layer, which means that the web-RTI functionality is implemented without changing any mechanism or source code of poRTIco. We keep in mind that, up to now, this work is still an on-going research project. The methodology presented still needs to be refined. As a result, the link, between the *WebServiceFederate* and the HLA federation, is not yet a completely strong and flexible link. The security issue is not fully addressed within the web service bridge. A solution envisaged is to use encapsulation and encoding method to ensure the security of the data package, but, the additional operation may cause another time delay issue. The authors are working hard on these gaps. In conclusion, we believe that the laboratory case study has proved the feasibility and efficiency of HLA Evolved WS methodology for testing enterprise data exchange configurations. A planned industrial case study will allow testing the proposed approach and improving its performance.

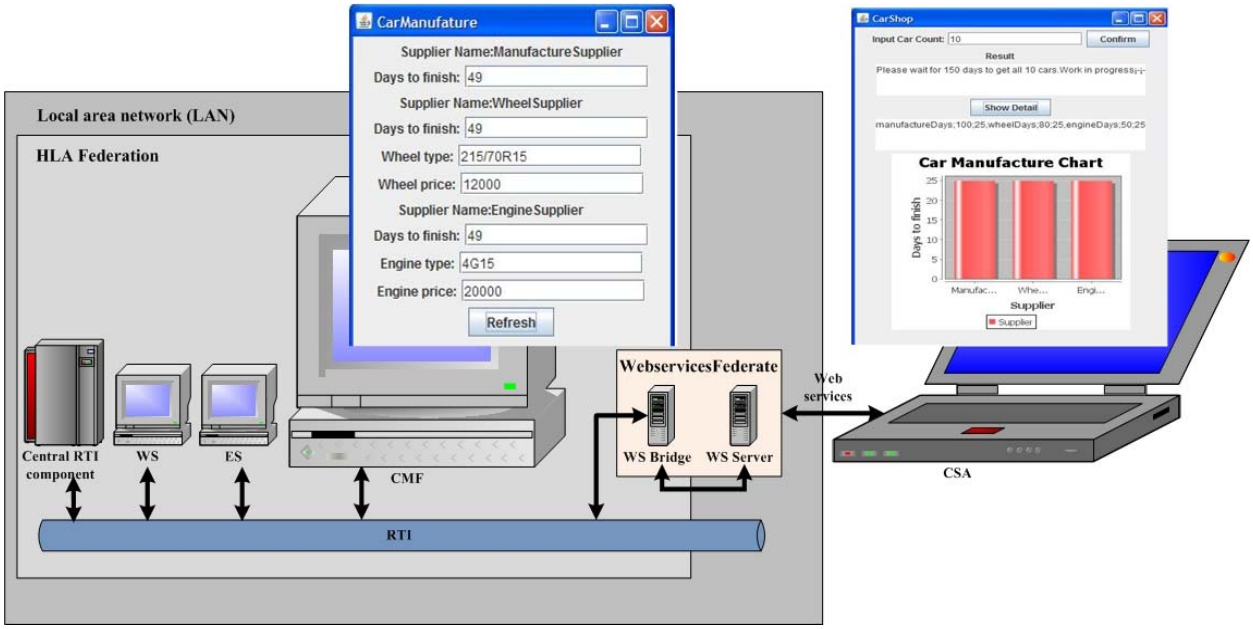

Figure 11: User interface of case study simulation

REFERENCES

Fujimoto, R. M. 2000. *Parallel and Distributed Simulation Systems*. New York: Wiley Interscience, Inc.

IEEE. 2000. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification*, Std 1516.2-2000. New York: Institute of Electrical and Electronics Engineers, Inc.

IEEE. 2003. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federation Development and Execution Process (FEDEP)*, Std 1516.3-2003. New York: Institute of Electrical and Electronics Engineers, Inc.

IEEE. 2010. *IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP)*, Std 1730-2010. New York: Institute of Electrical and Electronics Engineers, Inc.

Möller B., D. Clarence, and K. Mikael. 2007. "Developing Web Centric Federates and Federations using the HLA Evolved Web Services API." In *Proceedings of 2007 Spring Simulation Interoperability Workshop*, No. 07S-SIW-107, Norfolk, Virginia.

Möller, B., and S. Löf. 2005. "Mixing Service Oriented and High Level Architectures in Support of the GIG." In *Proceedings of the 2005 Spring Simulation Interoperability Workshop*, No. 05S-SIW-064, San Diego, California.

Möller, B., K. Morse, M. Lightner, R. Little, and R. Lutz. 2008. "HLA Evolved – A Summary of Major Technical Improvements". In *Proceedings of 2008 Euro Simulation Interoperability Workshop*, No. 08F-SIW-064, Edinburg, Scotland.

OMG. 2003. MDA Guide Version 1.0.1. Object Management Group, Document number: OMG / 20030601. Accessed March 30, 2011. www.omg.org/docs/-omg/03-06-01.pdf.

poRTIco. 2009. "Developer Documentation." Accessed March 30, 2011. http://porticoproject.org/index.php?title=Developer_Documentation.

Richardson, L., and S. Ruby. 2007. *RESTful web services.* Sebastopol, California: O'Reilly Media, Inc.

Tolk, A., and J. A. Muguira. 2004. "M&S within the Model Driven Architecture." *I/ITSEC*, Paper 1477, Orlando, FL.

Tu, Z., G. Zacharewicz, and D. Chen. 2010. "Harmonized and Reversible Development Framework for HLA Based Interoperable Application." *Proceedings of the International Conference on Modeling and Applied Simulation part of The 7th I3M*, edited by G. Bruzzone, C. Frydman, and M. A. Piera. Fes, Morrocco.

Zacharewicz G., C. Frydman, and N. Giambiasi. 2008. "G-DEVS/HLA Environment for Distributed Simulations of Workflows." *Simulation* 84(5): 197-213.

Zacharewicz, G., D. Chen, and B. Vallespir. 2008. "HLA Supported Federation Oriented Enterprise Interoperability, Application to Aerospace Enterprises." In *Proceedings of 2008 International Simulation Multiconference EuroSISO*, No. 08E-SIW-074, Edinburgh Scotland.

Zacharewicz, G., D. Chen, and B. Vallespir. 2009. "Short-Lived Ontology Approach for Agent/HLA Federated Enterprise Interoperability." In *Proceedings IEEE of International Conference I-ESA China 2009 Interoperability for Enterprise Software and Applications*, 329 – 335. Beijing, China: Institute of Electrical and Electronics Engineers, Inc.

AUTHORS BIOGRAPHIES

ZHIYING TU is a PhD student of University Bordeaux 1. His research subject is modeling and simulation of interoperable distributed enterprises. He recently focused on distributed simulation based on HLA and model reverse engineering. His e-mail address is zhiying.tu@ims-bordeaux.fr.

GREGORY ZACHAREWICZ is an Associate Professor in Bordeaux 1 University (IUT MP). His research interests include Discrete Event Modeling (e.g., DEVS), Distributed Simulation, Distributed Synchronization Algorithms, HLA and Workflow. He recently focused on Enterprise Modeling and Interoperability. He has published more than 20 papers in Conference and International Journals. He is Reviewer in Conferences (Summer SCS, WinterSim...) and SCS Simulation journals. His e-mail address is gregory.zacharewicz@ims-bordeaux.fr.

DAVID CHEN is a Professor at University Bordeaux 1. His research interests focus on enterprise modeling and interoperability. He has been actively participating in EU program (FP1-FP6), several cooperation programs between EU and China. David Chen is member of CEN TC 310/WG1 and ISO TC184/SC5/WG1 (Modeling and architecture) and involved in IFIP WG5.12 and IFAC WG5.3 (Enterprise Integration and Networking). He has published more than 80 papers in international journals and conferences. His e-mail address is david.chen@ims-bordeaux.fr.