

# USING WEB SERVICES AND BUSINESS PROCESS EXECUTION LANGUAGE FOR HLA-BASED DISTRIBUTED SUPPLY-CHAIN SIMULATION

*Malcolm Yoke Hean Low*

Singapore Institute of Manufacturing Technology  
71 Nanyang Drive, Singapore 638075  
yhlow@simtech.a-star.edu.sg

*Stephen John Turner*

School of Computer Engineering, Nanyang Technological University  
Nanyang Avenue, Singapore 639798  
ASSJTurner@ntu.edu.sg

## ABSTRACT:

*Supply-chains are large systems consisting of many components interacting in complex ways. The challenge faced by companies is how to design and manage such systems. Modeling and simulation enables analysis of complex systems but as the model increases in size and realism, a distribution capability is needed. While the High Level Architecture provides the infrastructure needed for large-scale distributed simulation, it does not have a mechanism to coordinate the configuration and invocation of geographically distributed models through a single point of access. This capability is provided by Web services. By publishing the HLA-based simulation models of their enterprise as Web services, companies can critically compare customized supply-chain scenarios by configuring, executing and analyzing the corresponding distributed simulations.*

*This paper describes an approach that uses Web services and Business Process Execution Language to support the business process (discovery, configuration, deployment and execution) of carrying out large-scale distributed simulations using the HLA framework.*

Keywords: Web Services, BPEL, SOAP, UDDI, Distributed Simulation, Interoperability

## 1. INTRODUCTION

Supply-chains are large systems consisting of many components interacting in complex ways. The challenge faced by companies is how to design and manage such systems. Supply-chain management covers the planning and management of material and information flow from the manufacturer, through the distributors, and finally to the customer. With the

globalization of markets, optimization of supply-chain management becomes more and more important.

Simulation of supply-chains can help in the optimization process by evaluating the impact of alternative supply-chain policies. Supply-chain simulation involves the simulation of the flow of materials and information through multiple stages of manufacturing, transportation and distribution. It includes the simulation of replenishments of inventory, the operations at each manufacturing stage, and shipments for the products from one stage to the next. A supply-chain often involves multiple companies across enterprise boundaries. Each of these companies may already have its own simulation model to perform “what-if” kind of analysis of its daily operations. Ideally, a supply-chain simulation can then be constructed by reusing these existing simulation models to increase the speed of modeling and analysis of alternative supply-chain scenarios [14].

However, companies participating in a supply-chain simulation may not be willing to share their models (and internal data) with other companies. In addition, existing simulation models may be implemented using different languages/packages and on different hardware platforms in each of these companies. Furthermore, with the globalization of markets, these companies may be located in different parts of the world. Thus, there is a need in supply-chain simulation for a common standard that enables the interoperability and distributed simulation of these geographically distributed simulation models through a well-defined interface.

The High Level Architecture (HLA), developed by the Department of Defense provides the infrastructure needed for large-scale distributed simulation. The HLA defines the rules and

specifications to support reusability and interoperability of different simulators [7]. In HLA terminology, a simulation component is referred to as a federate. A federation is then a set of federates working together to achieve a given goal. Each federate interacts with one another over the Runtime Infrastructure (RTI) [4]. A set of simulation models developed independently can be put together to form a larger simulation (or federation). Using the HLA, each participating federate in the federation can define the objects and interactions that are shared with others in its simulation object model (SOM), but its internal behavior (and data) is completely invisible to the outside world.

However, configuration and invocation of each simulation component under HLA still remains a manual process as there is no single point of access for HLA-based distributed simulation. HLA requires simulation users from each business entity to coordinate the configuration and invocation of simulation components. Thus, parameters of simulation components need to be communicated for each experiment. Furthermore, initiation of simulation components needs to be carried out physically at the simulation site. In a globalized market, where business partners can be located in various time zones, this is not a practical solution. The simplest solution to this problem will be to grant access to various simulation components through remote login. However, this would mean the need to open up the corporate network to outsiders, which is not acceptable to most corporations. Also, the list of partners in a supply-chain would be pre-determined before the simulation. Evaluating alternative supply-chain scenarios that involve simulation models from different partners cannot be easily carried out.

The aim of this project is to investigate the use of Web services and business process flow languages in providing a single point of entry to HLA-based distributed simulations. A framework that allows the user to discover, configure, modify, invoke and terminate supply-chain simulations through a web interface is developed. This framework is tested using a basic distributed supply-chain simulation model.

The rest of this paper is organized as follows. Section 2 gives some background information on Web services and the Business Process Execution Language. Section 3 gives a review on existing work relating to Web services and HLA-based distributed simulation. The distributed supply-chain simulation model developed in this study is described in Section 4. Section 5 describes the

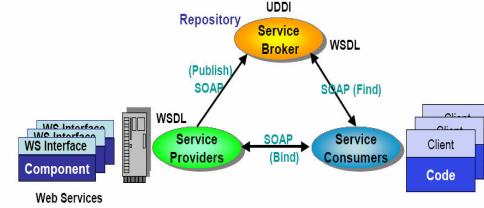


Figure 1: Components of Web Services

different Web services developed in this study. In Section 6, the Web service framework that allows the orchestration of the HLA-based distributed supply-chain simulation is described. Section 7 concludes the paper and outlines future work.

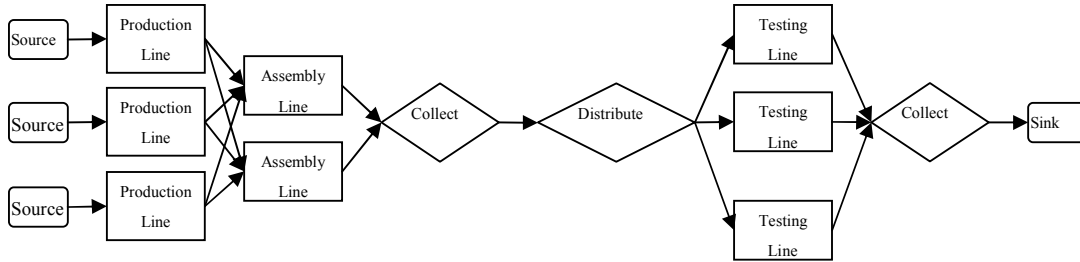
## 2. BACKGROUND

Web services technology can be used to provide a single point of access to HLA-compliant simulation components. Earlier attempts in distributed simulation modeling to establish interoperability with standards such as DCOM, CORBA, EJB and RMI had key limitations like platform dependency, tight coupling and limited interoperability. A Web service is a software component representing a specific set of business functions that can be described, published and invoked over the internet using open standards. By publishing simulation models of their enterprise as Web services, companies can critically compare customized supply-chain scenarios by configuring, executing and analyzing the corresponding distributed simulations.

Figure 1 shows the different components involved in Web services. Web services can be accessed using ubiquitous transport protocols like HTTP and SMTP and enable users to connect different components even across organizational boundaries in a platform- and language-independent manner. It supports XML based distributed computing using the following three industry standard technologies: 1) Web Service Description Language (WSDL); 2) Universal Description Discovery and Integration (UDDI); and 3) Simple Object Access Protocol (SOAP).

WSDL is an XML document that contains a set of definitions which describes what a Web service does, how it communicates, and where it is deployed. It provides all the information needed to access and use a Web service.

UDDI is a Web service registry that provides a mechanism to advertise and discover Web services. A UDDI registry contains categorized information about businesses and the services that



**Figure 2:** Configuration of a Supply-chain Federation

they offer, and it associates those services with technical specifications of the Web service. These technical specifications are usually defined using WSDL. A Web service consumer queries the UDDI registry to find the WSDL descriptions to determine how to use the Web service. The UDDI specification defines an API based on SOAP messages, with a WSDL description of the registry service.

SOAP is an extensible XML messaging protocol that forms the foundation for Web services. SOAP provides a simple and consistent mechanism that allows one application to send an XML message to another application. A SOAP message is a one-way transmission from a SOAP sender to a SOAP receiver, and any application can participate in an exchange as either sender or receiver. SOAP messages may be combined to support many communication behaviors, including request/response, solicit response, one-way asynchronous messaging, or event notification. SOAP messages can be exchanged over HTTP, JMS, or mail transport protocols. Currently the HTTP protocol is the most frequently used transport for SOAP messages.

While most Web services today are isolated and opaque, there is an increasing requirement to connect these Web services and specify how collections of Web services are jointly used to realize more complex functionality -- typically a business process.

BPEL (Business Process Execution Language) for Web services is an XML-based language that allows the specification of business processes and how they relate to Web services [8]. Written by developers from BEA Systems, IBM, and Microsoft, BPEL combines and replaces IBM's Web Services Flow Language (WSFL) and Microsoft's XLANG specification. BPEL specifies how a business process makes use of Web services to achieve its goal, as well as the Web services that are provided by a business process. A BPEL business process orchestrates

the Web services of its partners and supports the specification of business protocols between partners and views on complex internal business processes.

### 3. RELATED WORK

The issue of reusability of existing simulation models in a supply-chain simulation by interoperating different simulation components through the HLA has been studied in [1]. The study showed that considerable performance improvements can be achieved by fine-tuning the integration of the application with the HLA RTI. While the HLA provides support for reusability and interoperability of simulation models, it has no support for selective information hiding between groups of simulation components in a federation. To address the issue of protecting sensitive information within individual companies in the supply-chain, a hierarchical federation technique was developed in [2] that used a hybrid gateway/proxy approach to achieve selective information hiding. The study also demonstrated that the approach led to more loosely synchronized federates and resulted in better performance for the simulation.

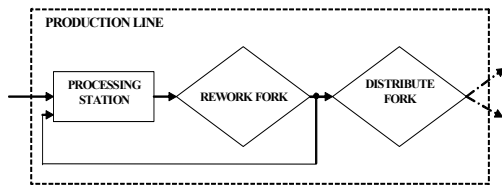
However, conventional HLA-based distributed simulation cannot operate across enterprise boundaries due to security restrictions such as firewalls that block non-standard ports and IP traffic. In one of the earliest work in the Extensible Modeling and Simulation Framework (XMSF), the issue of enabling a simulation to communicate with an HLA RTI through web-based services was investigated in [10]. While the web-enabled RTI enables interoperability of simulation federates through Web services, the discovery, configuration and invocation of the individual federates is still carried manually.

In another work under XMSF, the issue of using Web services and techniques such as the Model Driven Architecture (MDA) to facilitate design of meaningful interoperability among distributed and

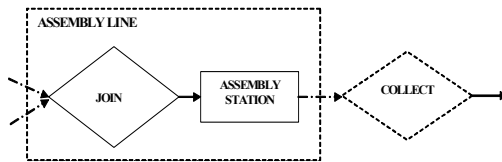
individually implemented simulation components are discussed [12]. The paper also discussed Grid services as a natural evolution step from Web services for implementing HLA-based distributed simulation. In [13], the implementation of a HLAGrid framework that supports HLA-based distributed simulation on the Grid platform is described. RTI services are exposed as Grid services, which provide more secure, scalable and coordinated management.

#### 4. SIMULATION MODEL

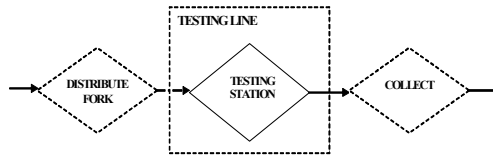
The case study used in this paper is a simulation model of a generic manufacturing process which consists of three stages: production, assembling and testing (see Figure 2). The same model was also used in [9] to study the performance of different parallel runtime systems for executing conservative parallel simulation.



The production stage consists of a number of production lines. Each production line produces a different component and receives its input (raw materials) from a **source node**. Each production line is made up of a sequence of **processing station** and **rework fork**, followed by a **distribute fork**. Products from the output of a processing station will be sent back for rework via the rework fork, or distributed uniformly to the assembly lines.



The assembly stage consists of a number of assembly lines. Each assembly line is made up of a **join node** and an **assembly station**. The join node collects parts from all the available production lines. The assembly station assembles the different parts into the finished product. The finished products from all the assembly stations are sent to a **collect node** to queue for testing.



The test stage consists of a number of **testing stations**. The assembled products from the collect node are distributed to each testing station uniformly. The final tested products from each testing station are collected by another **collect node** and sent to the **sink node**.

Each of the three types of stations (processing, assembly and test) has a mean service time. The other four queue nodes (rework fork, distribute fork, join node and collect node) do not advance simulation time. The three stations, four queuing nodes and the source and sink nodes, are modeled as individual federates with configurable attributes. The simulation model can be generated using the following parameters:

##### 1. Supply-chain (Federation) parameters

- number of production lines
- number of processing stations per production line
- number of assembly stations
- number of testing stations

##### 2. Supply-chain Node (Federate) parameters

- average release time of events by source node
- mean service time of processing station
- mean service time of assembly station
- mean service time of testing station
- rework probability of re-work fork

Federate Name	Parameters
Sinkfed	fName, inFName, numFeds, simEndTime
Sourcefed	fName, outFName, maxItem, interArrivalTime
PSfed	fName, outFName, inFName[], avgServiceTime
ASfed	fName, outFName, inFName, avgServiceTime
TSfed	fName, outFName, inFName, avgServiceTime
ReworkForkfed	fName, outFName[], inFName, reworkProb
DistributeForkfed	fName, numOutputs, outFName[], inFName
Joinfed	fName, outFName, numInputs, inFName[]
Collectfed	fName, outFName, numInputs, inFName[]

**Table 1:** Input Parameters for Supply-chain Federates

Each of the federates is invoked through command line parameter input. Table 1 shows the parameters for the different federates. Each of the federates is identified by its federate name (fName). The sink federate acts as the controller in the federation and has parameters on its input

federate (inFName), the total number of federates to synchronize (numFeds) and the simulation end time. The source federate has parameters on its output federate (outFName), the maximum number of items to generate (maxItem) and the inter-arrival time of the items. The processing, assembly and test stations (PSfed, ASfed and TSfed), each have parameters on their output federate, input federate and the average service time. Note that the PSfed can have multiple input federates (denoted by an array []) due to items arriving from rework. The rework federate is associated with a rework probability parameter. The distribute, join and collect federates have parameters on the number of outputs and inputs respectively.

The sink node is assigned to act as the manager federate which creates the federation and carries out synchronization for all federates. Each federate carries out the following tasks during its execution:

1. Join the federation
2. Initialize the federate
3. Wait for synchronization
4. Run the main simulation
5. Write output results on file
6. Wait for synchronization
7. Resign federate from federation

The sink federate has the additional task of creating the federation before step 1, maintaining the synchronization point at steps 3 and 6, as well as destroying the federation after step 7.

## 5. WEB SERVICES

In order to demonstrate the orchestration of Web services using BPEL, the different components required to set up a HLA-based distributed simulation are first exposed as Web services. These components include different federates of the supply-chain simulation model described in section 4 and a service for generating the federation object model (FOM) for the supply-chain federation. In this section, the different services exposed by these components will be described.

### 5.1 FEDERATE WEB SERVICES

Each of the federates in the supply-chain model is exposed as a Web service which consists of the following three methods:

- **invoke()**: The federate can be executed by calling the invoke() method. This will create a separate thread via a Web service using the

Java Runtime class (see Figure 3) for the associated federate. Objects instantiated from the Java Runtime class take a command line as the input, and spawn off a process to execute the command line.

- **uploadSOMFile()**: In HLA-based simulation, the SOM of each federate defines the data that it can produce or consume; while the FOM defines the data that can be exchanged between federates in a federation. The FOM is made up of SOMs from individual federates in the federation. To generate the FOM, a file representing the SOM of the federate will be uploaded to the URL specified in the input parameter.
- **uploadResults()**: The simulation result of the federate will be uploaded to the URL specified in the input parameter.

The sink federate has an additional Web service **simEndStatus()** that reports if the distributed simulation has completed. All the Web services are deployed using Apache AXIS (Apache eXtensible Interaction System) which provides an easy method to deploy Java programs as Web services. Figure 3 shows an example of how a federate can be exposed as a Web service.

```
import java.lang.*;

public class <federate>WebService {
    // execute the federate
    public String invoke()
    {
        .. . . .
        try
        {
            Process p =
                Runtime.getRuntime().exec("java
                <federate><parameters>");
        }
        catch (Exception e) { }
        .. . . .
    }

    // upload SOM file
    public String uploadSOMFile(String url) {
        .. . . .
    }

    // upload simulation result
    public String uploadResults(String url) {
        .. . . .
    }
}
```

**Figure 3:** Example of a Federate Exposed as a Web Service

### 5.2 FOM GENERATOR WEB SERVICE

Each federate in the supply-chain federation may have a different SOM. These SOMs contain the object and interaction classes which the federate will use to communicate with each other through the RTI. Depending on the configuration of the

supply-chain, the FOM may be different for different supply-chain federations. A FOM must be generated from all the federates' SOMs in order to represent a common set of object and interaction classes in the federation.

This service is provided by the FOM generator Web service. The FOM generator has the following two functions:

```
public String downloadFile(String fileURL);
public String runGen(String outputFileName);
```

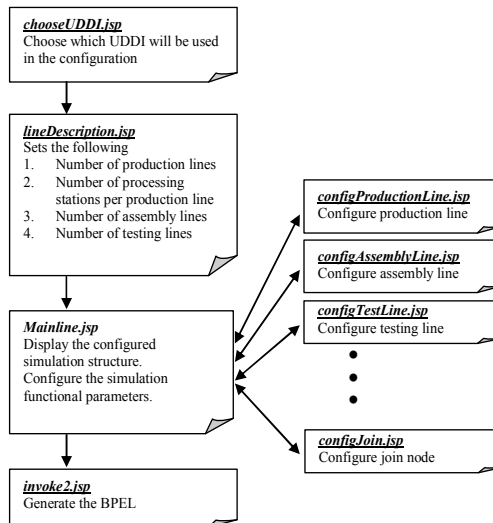
The `downloadFile()` function takes the URL of the SOM of each federate as a parameter and makes a copy of the SOM located at the URL.

The `runGen()` function takes as parameter the desired name of the FOM (final integrated fed file). It parses through all the downloaded SOM files and generates a fed file with the union of all user object classes and the union of all user interaction classes. For the purpose of this prototype, the rest of the pre-defined classes and pre-defined attributes are assumed to be the same for all federates. This generated fed file is the FOM for the federation. The two methods of the FOM generator are also exposed as Web services using Apache AXIS.

## 6. BUSINESS PROCESSES

A dynamic reconfigurable business process for a HLA-based distributed supply-chain simulation involves the design, discovery, configuration, deployment, invocation and analysis of the supply-chain components. In this prototype, the Business Process Execution Language (BPEL) is used to enable these requirements. BPEL allows the automatic orchestration, discovery, and matching of business process flow using the available Web services of simulation federates. It allows the user to connect these simulation federates' Web services and specifies how these Web services are jointly used to realize the supply-chain federation. The supply-chain federation is thus not restricted to a particular flow, but can dynamically bind different federate Web services of different partners to form new business process flows.

A web user interface is developed to allow the user to configure, deploy, invoke the simulation and collect the simulation result. As the simulation configuration may be different for each run, static BPEL cannot be used and a new



**Figure 4:** Web Interface for Simulation Configuration and BPEL Generation

BPEL must be generated for each new run. The BPEL generator and the federate Web services are transparent to user. The web interface guides the user through the simulation process. There are four steps in the business process.

1. Simulation configuration and BPEL generation
2. BPEL deployment
3. BPEL invocation
4. Simulation Result Collection and Termination

### 6.1 SIMULATION CONFIGURATION AND BPEL GENERATION

A set of web pages is implemented for the configuration of the simulation model and invocation of the BPEL (see Figure 4). The user first chooses which UDDI will be used to search and discover the Web services. A UDDI server is hosted locally for testing purposes in this prototype.

Once the UDDI server is selected, the user can specify the number of production lines, assembly lines and testing lines. Once the supply-chain configuration is fixed, the user will have to select a Web service for each of the federates in the supply-chain. For each of the federates in the supply-chain, the user will be presented with a list of Web services matching the type of federate (which may be provided by different business partners). This list of Web services is obtained by querying the selected UDDI with the respective NAICS index for the federate. NAICS, or the North American Industry Classification System, is a commonly used categorization scheme to

classify businesses and services in a standard, universally-known way.

The registration of the Web services to the UDDI will usually be carried out by the federate web service providers after the Web services are deployed. As the UDDI is originally designed for commercial usage, there is no NAICS category for simulation model Web services. In this project, an extension of the NAICS classification is introduced for simulation federate Web services by creating new categories for the nine types of federates. The new category names and indexes are listed in Table 2.

Federate name	NAICS classification index	NAICS classification name
Source	200401	Source federate
Production station	200402	Production station federate
Rework fork	200403	Rework fork federate
Distribute fork	200404	Distribute fork federate
Collect	200405	Collect federate
Assembly station	200406	Assembly station federate
Testing station	200407	Testing station federate
Join	200408	Join federate
Sink	200409	Sink federate

**Table 2:** UDDI Extension Definition

The BPEL generator is used to generate the BPEL for simulation configuration and invocation. When the user passes the simulation configuration into the BPEL generator, the BPEL generator will generate the BPEL file together with the set of configuration files required by the Collaxa server (see section 6.2). The generated BPEL file will coordinate the entire configuration and invocation phase of the simulation and also monitor the result collection phase. The generated BPEL file is based on the specification of BPEL version 1.1. Figure 5 shows the outline of the BPEL file generated by the BPEL generator.

A BPEL file consists of four parts. The first part contains the URLs of the Web services partners, which includes the simulation federates' Web services endpoints, and the endpoint of the FOM generator. The second part is the partnerLink definition. Each partnerLink in the BPEL represents one simulation federate Web service. The services' URLs are referenced when declaring the partnerLink. The third part is the message definition. All the SOAP messages that are used in the configuration and invocation process are declared in this section. The final part is the actual configuration and invocation process. The invocation process can be further separated

```

<process name="Flow"
  targetNamespace="http://samples.cxoh.com"
  xmlns:tns="http://samples.cxoh.com"
  xmlns:odpa="http://pdp32.8080/axis/forngenWS/jws"
  ...
  >
  Part 1

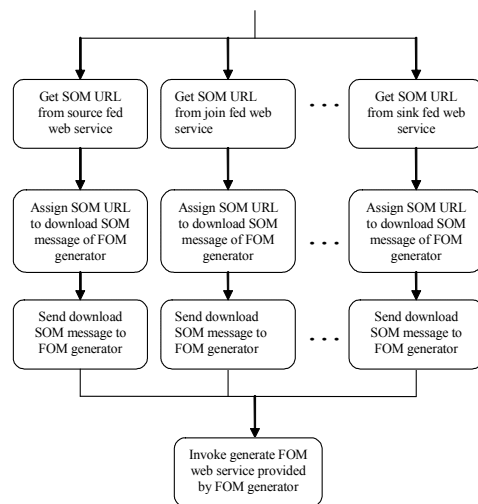
  <partnerLinks>
  <partnerLink name="client"
    partnerLinkType="tns:Flow"
    myRole="FlowService"
    partnerRole="FlowRequester"/>
    ... more partnerLinks ...
  </partnerLinks>
  Part 2

  <variables>
  <variable name="Input" messageType="tns:InitiateFlowSoapRequest"/>
  ... more variables ...
  </variables>
  Part 3

  <sequence name="sequence">
  ... Web Services interaction sequence...
  </sequence>
  Part 4

```

**Figure 5:** BPEL File Skeleton for Supply Chain Distributed Simulation



**Figure 6:** BPEL: FOM Generation Section

into three sections, FOM generation, simulation invocation, and result collection.

Figure 6 shows the flow of the Web services in the FOM generation section of the BPEL generated by the BPEL generator. As shown in the figure, the upload of the SOM for each federate can be carried out in parallel. For this section, the BPEL generator will make use of the “Flow” construct in BPEL to achieve concurrency and synchronization. The three tasks in gathering the SOM from each federate is specified within the “Sequence” construct of BPEL that allows the execution of the three tasks sequentially.

```

...
<flow standard attributes>
  <sequence> get SOM for source fed</sequence>
  ...
  <sequence> get SOM for sink fed</sequence>
</flow>
...

```

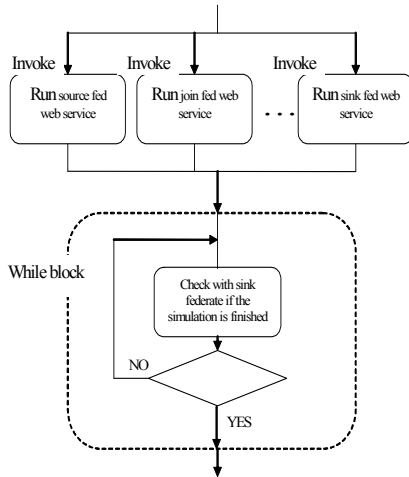


Figure 7: BPEL: Invocation Section

Figure 7 shows the flow of the Web services in the invocation section of the BPEL. Each federate is invoked in parallel with the parameters provided by the user through the web interface. As the sink federate acts as the manager federate in the federation, it has an additional Web service that returns the termination status of the federation. The BPEL generated makes use of the “While” construct to poll the sink federate Web service to determine when the supply-chain simulation has completed. The result collection section of the BPEL has a similar structure to the FOM generation and will not be described in this paper.

## 6.2 BPEL DEPLOYMENT, INVOCATION AND TERMINATION

Once the BPEL file is generated, it is deployed to the Collaxa server. The Collaxa BPEL Orchestration Server is a scalable and easy-to-deploy infrastructure for modeling, connecting, deploying and managing BPEL processes [3]. Collaxa was purchased by Oracle in 2004, which now offers Oracle BPEL Process Manager as a product [11]. Software providers such as IBM also have corresponding Eclipse-based development tools for BPEL, such as the WBI Modeler [5] and WebSphere Studio Application Developer Integration Edition [6].

The entire BPEL process is also deployed as a Web service. It has its own WSDL and waits for an incoming SOAP message to start. The invocation web page embeds java codes that can send a SOAP invocation message to the BPEL server. For evaluating the prototype, one computer was used to run the rtiexec (DMSO RTII.3NG-V6). To invoke the distributed supply-

chain simulation, a SOAP message is sent from the web portal server to the BPEL server to start up the BPEL process. The BPEL process then configures and invokes the simulation. After the simulation is completed, the simulation results will be collected by the BPEL service automatically and made available for analysis by the user through the web interface.

## 7. CONCLUSION

In this paper, an approach for orchestrating distributed supply-chain simulations through Web services and BPEL is studied. Our prototype shows that supply-chain federates that are exposed as Web services can be orchestrated based on a business process specified by a BPEL file. The BPEL file can then be deployed as a Web service that will invoke the chain of Web services that configures and executes the supply-chain distributed simulation.

While the FOM generator creates the FOM based on the union of all the SOMs collected from the different federate Web services, it does not check if the supply-chain process flow/semantics is correct. For example, a user can configure the simulation to connect the Sink to the Processing Station component to produce an invalid supply-chain model. Further work should incorporate semantic checking in the FOM generator to avoid invalid configurations.

Future work to this project will include investigating extending the approach to encompass the entire business process life-cycle from existing process of design, discovery, configuration, deployment, execution and analysis, to include a feedback mechanism that automatically re-structures the business process flow based on the analysis results and re-deploys the new business process. This work will be carried out jointly through the Shared University Research (SUR) grant that was awarded by IBM to the Parallel and Distributed Computing Center in the Nanyang Technological University and the Singapore Institute of Manufacturing Technology.

## ACKNOWLEDGMENTS

The authors would like to thank Cai Ping and Shakti Singh Butola for implementing the prototype for this project.

## REFERENCES

- [1] Cai, W., Turner, S.J., and Gan, B.P. Adapting a Supply-Chain Simulation for

- HLA. In *Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT'00)*, pp. 71-78, 2000.
- [2] Cai, W., Turner, S.J., and Gan, B.P. Hierarchical Federations: An Architecture for Information Hiding. In *Proceedings of the Fifteenth Workshop on Parallel and Distributed Simulation*, pp. 67-74, 2001.
- [3] Collaxa Inc. Collaxa: Model, Deploy and Manage BPEL Processes. <http://www.collaxa.com>, 2002.
- [4] DMSO. RTI 1.3-Next Generation Programmer's Guide Version 5, DoD, DMSO, 2002.
- [5] IBM. WebSphere Business Modeler. <http://www-306.ibm.com/software/integration/wbimodeler/>, 2005.
- [6] IBM. WebSphere Studio Application Developer Integration Edition. <http://www-306.ibm.com/software/integration/wsadie/>, 2005.
- [7] Kuhl, F., Weatherly, R., and Dahmann, J., *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, Prentice Hall PTR, 1999.
- [8] Leymann, F. and Roller, D. Processes in a Web Services world. <http://www-128.ibm.com/developerworks/webservices/library/ws-bpelwp/>, 2002.
- [9] Lim, C.-C., Low, Y.-H., Cai, W., Hsu, W.J., Huang, S.Y., and Turner, S.J. An empirical comparison of runtime systems for conservative parallel simulation. In *2nd Workshop on Runtime Systems for Parallel Programming (RTSPP'98)*, pp. 123-134, 1998.
- [10] Morse, K.L., Drake, D.L., and Brunton, R.P.Z. Web Enabling HLA Compliant Simulations to Support Network Centric Applications. In *2004 Command and Control Research and Technology Symposium*, paper 172, 2004.
- [11] Oracle. Oracle BPEL Process Manager. <http://www.oracle.com/technology/products/ias/bpel>. 2005.
- [12] Pullen, J.M., Brunton, R., Brutzman, D., Drake, D., Hieb, M., Morse, K.L., and Tolk, A. Using Web Services to Integrate Heterogeneous Simulations in a Grid Environment. *Future Generation Computer Systems*, 21(1), pp. 97-106, 2005.
- [13] Xie, Y., Teo, Y.M., Cai, W., and Turner, S.J. Resource Provisioning for HLA-based Distributed Simulation on the Grid. In *Proceedings of the 19th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation*, pp. 282-291, 2005.
- [14] Zee, D.J. and Vorst, J. A Modeling Framework for Supply Chain Simulation: Opportunities for Improved Decision Making. *Decision Sciences*, 36(1), pp. 65-93, 2005.

#### AUTHOR BIOGRAPHIES

**MALCOLOM YOKE HEAN LOW** is a Research Engineer with the Planning and Operations Management Group at the Singapore Institute of Manufacturing Technology. He received his doctorate from Oxford University in 2002. His research interests are in the areas of adaptive tuning and load-balancing for parallel and distributed simulation systems, and the application of multi-agent technology in supply chain logistics coordination.

**STEPHEN JOHN TURNER** joined Nanyang Technological University (Singapore) in 1999 and is currently an Associate Professor in the School of Computer Engineering and Director of the Parallel and Distributed Computing Centre. Previously, he was a Senior Lecturer in Computer Science at Exeter University (UK). He received his MA in Mathematics and Computer Science from Cambridge University (UK) and his MSc and PhD in Computer Science from Manchester University (UK). His current research interests include: parallel and distributed simulation, distributed virtual environments, grid computing and multiagent systems.