Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

WEB SERVICE TECHNOLOGIES AND THEIR SYNERGY WITH SIMULATION

Senthilanand Chandrasekaran
Gregory Silver
John A. Miller
Jorge Cardoso
Amit P. Sheth

Department of Computer Science/LSDIS Lab
The University of Georgia
Athens, GA 30602-7404, U.S.A.

ABSTRACT

The World Wide Web has had an huge influence on the computing field in general as well as simulation in particular (*e.g.*, Web-Based Simulation). A new wave of development based upon XML has started. Two of the most interesting aspects of this development are the Semantic Web and Web Services. This paper examines the synergy between Web service technology and simulation. In one direction, Web service processes can be simulated for the purpose of correcting/improving the design. In the other direction, simulation models/components can be built out of Web services. Work on seamlessly using simulation as a part of Web service composition and process design, as well as on using Web services to re-build the JSIM Web-based simulation environment is highlighted.

1 INTRODUCTION

The World Wide Web has had an huge influence on the computing field in general as well as simulation in particular (*e.g.*, Web-Based Simulation). A new wave of development based upon eXtensible Markup Language has started. Two of the most interesting aspects of this development are the Semantic Web and Web Services.

This paper examines the synergy between Web service technology and simulation. In one direction, Web service processes can be simulated for the purpose of correcting/improving the design or even for making adaptive changes at runtime (Miller *et al.* 2002). The success of an organization depends greatly on the efficiency and effectiveness of its business processes. The advent of Web services and Web processes (composition of Web services) enables organizations to easily collaborate in their business processes.

When composing a Web process it is useful to analyze and compute overall operational properties. This allows organizations to translate their vision into their business processes more efficiently, since Web process can be designed according to operational metrics. Operational metrics can be described using a suitable Quality of Service (QoS) model (Cardoso, Miller *et al.* 2002; Cardoso, Sheth *et al.* 2002; Miller *et al.* 2002). Such a model makes possible the description of Web services and Web processes according to their timeliness, cost of service, and reliability.

QoS analysis becomes increasingly more important when Web processes model complex and mission-critical applications. QoS analysis and monitoring serve to ensure that each application meets user requirements. For e-commerce processes, it is important to know the QoS an application will exhibit before making the service available to customers. At runtime, it is important to identify whether processes exhibit the metrics specified at design time. If threshold levels are reached, adaptation strategies need to be applied to correct the operational metrics of Web processes.

The analysis of Web processes according to their QoS can be carried out using several methods. While mathematical methods have been effectively used (Cardoso, Miller *et al.* 2002), another alternative is to utilize simulation analysis (Miller *et al.* 2002). Simulation plays an important role by exploring "what-if" questions during the process composition phase. Our earlier work on workflows and simulation (Miller *et al.* 2002) enables us to perceive how simulation can serve as a tool for the Web process composition problem. The analysis of the QoS of Web processes differs from the analysis of workflows due to the distribution, autonomy, and heterogeneity of its components.

Our current work on using simulation for Web services focuses on extending JSIM (Miller *et al.* 1997; Nair *et al.* 1996) and integrating it with Web process design tools, as well as Web process enactment engines. The designer,

Web Process Design Tool (WPDT), allows composition to be done graphically. To aid the user in this composition task, our system is enhanced with enactment and simulation features. Enactment of a process helps in evaluating the performance of the individual services and simulation is done to study the process in action, before enactment.

In the other direction, using Web services for simulation, simulation models/components can be built out of Web services. Well tested simulation models may be placed on the Web for others to use. Resources and tools used in simulation environments make excellent candidates for Web services. If this vision can be realized, future development can be done on a higher plane, allowing better and more comprehensive solutions to be developed.

The rest of the paper is organized as follows. In Section 2, we present the related work in this area while Section 3 introduces composite Web services, issues related to specification of Web services, and composition of Web services. Section 4 covers our system architecture, Web process designer tool, and our enactment technique. In Section 5, we explain our performance evaluation approach for evaluating/comparing the invoked Web services. Simulation and its application to Web process composition are discussed in Section 6. Section 7 discusses how simulations may be built out of Web services. Finally, conclusion and future work are presented in Section 8.

2 RELATED WORK

Some work has begun on the use of simulation to study Web service composition and Web processes, but little work has been done on the use of Web services to build simulation environments. Web service composition is an active area of research, with many concepts and languages being proposed by different research groups. IBM has proposed WSFL (Web Service Flow Language) (Leymann 2001), an XML based language developed to describe complex service compositions. WSFL supports a flow model and a global model specification for each Web process. The flow model defines the structure of the Web process, while the global model specifies the Web services, which implement the activities in the process. Microsoft's Web service composition language, XLANG (Thatte 2001), extends the WSDL (Web Service Description Language) (Christensen *et al.* 2001) to provide a model for orchestration of services. XL (Florescu *et al.* 2002), another portable W3C compliant XML programming language, is designed for the implementation of Web Services. In contrast to these XML based standards, researchers are developing DAML-S (Ankolekar *et al.* 2001), which aims to automate Web service tasks (discovery, composition, invocation, and monitoring) using specifications based on ontologies. DAML-S, unlike the earlier XML based languages, is capable of describing the semantics of Web services. Issues such as searching for services and interoperability of selected services arise when a Web service composition is done. Cardoso and Sheth (2002) explore semantic searching for Web services and their interoperability. An ontology based solution is proposed in that paper.

Though use of simulation to test processes has been carried out earlier for workflow models (Miller *et al.* 1995; Miller *et al.* 2002), simulation of composite Web services represents a new direction. The work that most closely relates to ours is described in Narayanan and Mcllraith (2002). In their work, DAML-S service descriptions of composite services are encoded in a Petri Net formalism, providing decision procedures for Web services simulation, verification, and composition.

3 COMPOSITE WEB SERVICES

Earlier attempts in distributed computing to establish interoperability with standards such as DCOM, CORBA, EJB and RMI had key limitations like platform dependency, tight coupling and limited interoperability. Hence the industry saw the need for a new distributed computing approach that can overcome these limitations. The Web services paradigm has been proposed to solve these problems.

A Web service is a universally accessible software component deployed on the Web. Such a software component is described by an interface listing the collection of operations that can be performed on it. Web services (unlike earlier distributed computing models) are suitable for integrating e-business applications for the following reasons. They are XML based, allowing them to address data encoding problems that existed in earlier models. They support XML based distributed computing using SOAP (Simple Object Access Protocol) and can be accessed using ubiquitous transport protocols like HTTP and SMTP.

Individual (atomic) Web services provide only specific functionality. A Web Process (composite Web service) is a service made up of several components, each of which may be an atomic Web service. A process is created by orchestrating existing Web services, and defining the control and data flows among them.

A proper specification of Web services and Web processes is required for efficient inter-operation in a distributed environment.

3.1 Web Services Specification

Description of services in a widely accepted format is vital for the widespread use of Web services. Service providers describe their Web services and advertise them in a registry. This enables service requesters to search for services, that match their requirements. XML, the emerging standard for data representation, has been chosen as the language for describing Web services. The specification of a Web service,

should include syntactic (what does it look like), semantic (what does it mean) and QoS (how well does it perform) information. Quality of Service (QoS) (Cardoso, Sheth *et al.* 2002) attributes, which are timeliness, cost of service, and reliability, provide a description of the quality that can be expected from the service. Time, cost and reliability are some of the QoS attributes that describe a service.

WSDL and DAML-S are the two major languages used to describe Web services. WSDL is the W3C standard XML language used to specify a Web service's interface and it defines the syntactic information about a service. DAML-S is an ontology based interface description language, that can describe the syntactic as well as the semantic content of a service. DAML-S describes some nonfunctional QoS related attributes of a service, but WSDL does not provide any QoS information.

3.2 Web Process Specification

A Web Process needs to be described in a way similar to the way a Web service's interface is described. Popular languages for describing the composition of Web services include WSFL, XLANG, and DAML-S. These languages can be used to describe composed process. Interoperability issues among the chosen Web services needs to be taken care of when using these languages to develop a process. The developer has to explicitly understand the details of the interfaces and specify the mappings that are required. As WSFL is one of the mature and practical languages for Web process composition, we chose WSFL to represent composed processes in our system. The WSFL specification is currently being worked on to add QoS extensibility elements. Since this work is still in progress, we have extended WSFL's specification to include time, cost, and reliability QoS attributes for each activity in our system.

3.2.1 Web Service Composition

Web service composition is the creation of a Web Process from individual Web services. Web processes facilitate expanding the utility of Web services. Web service composition can be either static or dynamic. In static composition, the services are predetermined during the design of the Web process. In a dynamic composition, the Web service to be used for an activity is decided at run-time by, for example, the process enactment engine. Dynamic composition involves run-time searching of registries to find services.

Web service composition can be represented as a workflow graph with activities (services) and transition links (control and data). In composing Web services to form processes, data links and control links are used to specify the data flow and control flow respectively among the services. Standard constructs like XOR splits, AND splits, XOR joins, AND joins are used to capture the execution logic in the process. An XOR split is used to indicate the branching of the control flow in one of the outgoing control links. An AND split indicates the branching of the control flow in all of the outgoing control links in parallel. An AND join indicates synchronizing on all incoming controls links while an XOR join indicates waiting on one of the indicated incoming control links.

3.2.2 Scenario

Figure 2 depicts the tasks involved in buying a book. The activities (Web services) in this process are *SearchAmazonCatalog*, *ChooseProduct*, *CheckCredit*, *CheckInventory*, *GenerateBackOrder*, *ReleaseOrder* and *SendCreditLowInfo*. Information about related books for a given search is retrieved using the SearchAmazonCatalog service and a book is chosen by the user via the ChooseProduct service. The user's account is then checked for sufficient funds using the *CheckCredit* service. *CheckCredit* service is an example of an XOR split activity. After the *CheckCredit* service, the control flows in one of the two control links depending on whether the *CheckCredit* service returns success or failure. If the user has sufficient credit, the *CheckInventory* service is invoked; else the *SendCreditLowInfo* service is invoked. If the *CheckInventory* Web service returns true the *ReleaseOrder* service is invoked to send the books, else the *GenerateBackOrder* service is invoked.

4 SYSTEM DESIGN

In this section, we describe our system architecture for designing, simulating, and creating Web service processes (see Figure 1).

A *Web Process Designer Tool* (WPDT) is used to compose processes and store the designs in a repository. These Web process models are transformed automatically to JSIM simulation models using the inbuilt *Simulation Model Generator*. Simulation is done to evaluate the performance of the process. The simulation model can be modified to answer "what-if" questions about the process. The Web process may also be enacted on a test basis, to evaluate the performance of its Web services. Based on the results of the simulation and test enactment, we may adapt the process to meet our needs.

4.1 WPDT - Web Process Design

As part of our work, we have developed the WPDT, for composing Web services into Web processes (see Figure 2). WPDT is a process-design tool, that allows static composition of Web services to build Web processes. WPDT stores Web processes as WSFL specifications.

We will briefly explain how WPDT is used to design Web processes. A Web process, similar to a workflow, is

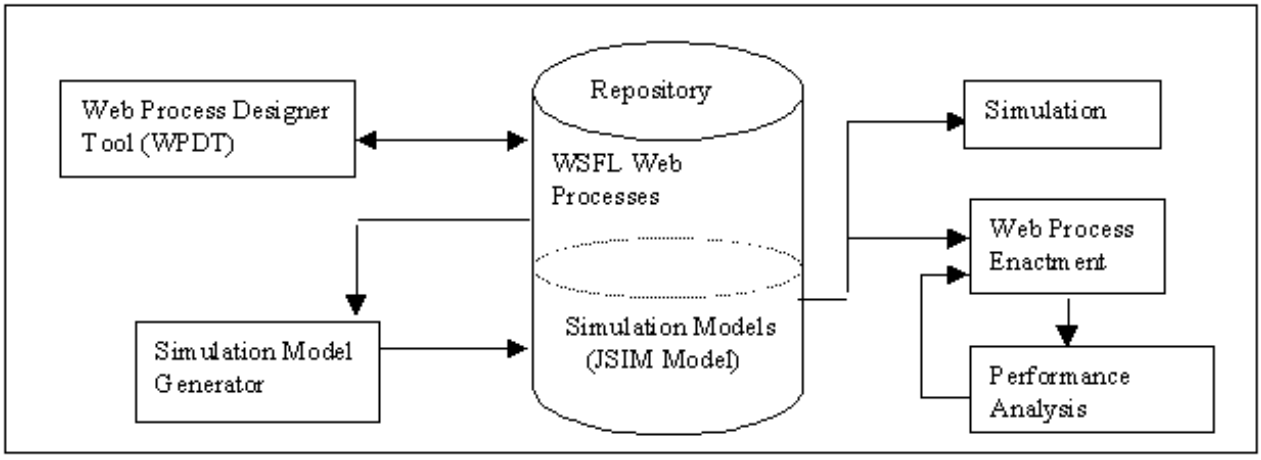

Figure 1: System Architecture

represented as a digraph with source, sink, activities (Web services) and transition links. In WPDT, we have three kinds of nodes, namely, source node, sink node and activity node. Users need to provide the information about the Web services implementing each activity. This includes the WSDL file location and QoS information. The data flow (DataLink) between the activities is represented as green transition links (not in picture). The black transition links represents both data flow and control flow (DataLink / ControlLink) between the activities as shown in Figure 2.

WPDT stores its Web process designs (XML based WSFL specifications) in a Db4XML repository (Sipani *et al.* 2002), which is an XML database, developed at the University of Georgia. Since Db4XML supports XQuery (XML Query Language), users can compose a Web process and efficiently query the design using the XQuery language. Efficient querying of these Web process designs is desired when one needs to extract information about a design.

To test a process, we need to enact it. In the next section, we discuss enactment strategies that can be employed to invoke a Web process and our system implementation.

4.2 Web Process Enactment

Web process enactment is similar to a workflow enactment, the difference being the components of a workflow are activities while the components of a Web process are Web services. Web services differ from workflow activities in their distribution, autonomy and heterogeneity. Substantial research on workflow enactment, has been done in the LSDIS Lab at the University of Georgia (Sheth *et al.* 1996; Miller, Palaniswami *et al.* 1998, Kochut *et al.* 1999). Based on our work, we put forward two approaches for enacting Web processes: a centralized approach and a distributed approach.

The centralized approach is based on a client/server architecture. It uses a controller, which controls the execution of the Web process and serves as the client requesting service from the Web services. The controller invokes a Web service, gets the results, based on the results and the Web process design specification, the controller then invokes the next appropriate Web service. This controller-based approach is the easiest means of enacting Web processes.

The distributed approach for a Web process enactment is more complex. In a distributed approach there is no controller, and Web services are expected to share the execution context of the process, so that distributed execution is possible with the collaboration of other Web services. This sharing of the context can be achieved dynamically by peer-to-peer communication between Web service hosts (Benatallah *et al.* 2002), or using agent based solutions (Stormer 2001).

We have implemented the centralized technique in our system with a controller executing the composed Web process. In our implementation, a Perl controller module, manages the entire Web process execution. This Perl enactment code is generated from the WSFL specification of the Web process. Perl was selected because its easy to use SOAP modules help in quickly scripting the process description from the WSFL specification.

During the test enactment of a Web process, we instrument our controller module to determine the empirical data associated with the test. This includes measuring the total time taken for each Web service invocation. This enables us to analyze individual Web service performance and analyze the distribution of service times associated with its execution.

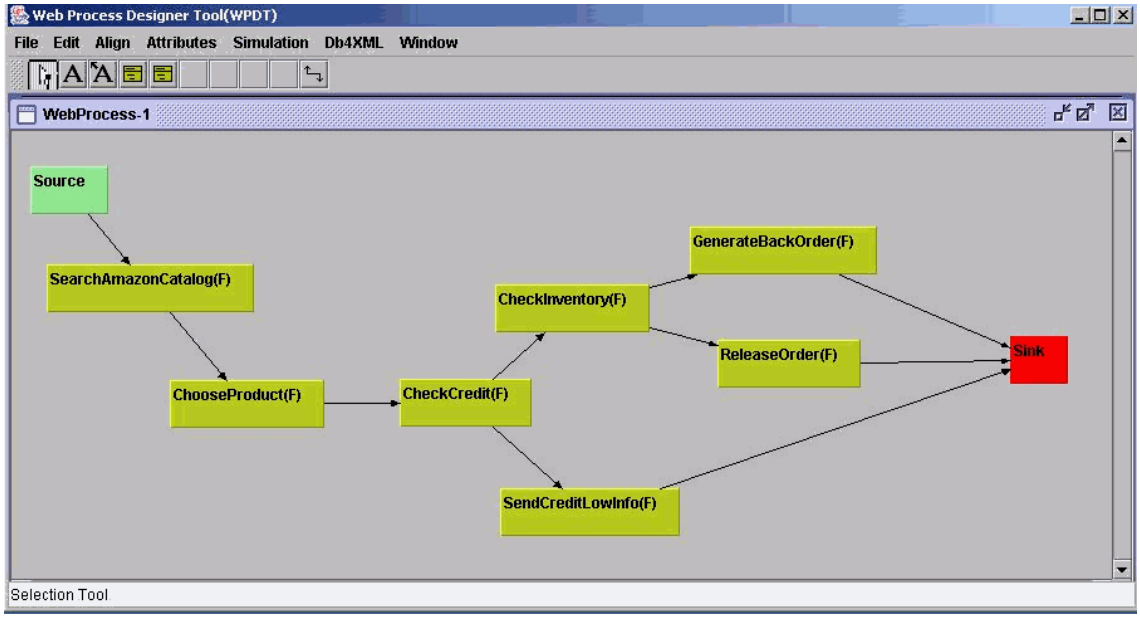

Figure 2: Web Process Design using WPDT

5 PERFORMANCE EVALUATION

Performance evaluation of Web services can help implementers understand the behavior of the activities in a composed process. Since the performance of a single Web service has the potential to affect the performance of an entire Web process, it is wise to evaluate the performance of the critical services within a process before enactment.

The time taken by a single Web service depends on *Service time* (S), *Message delay time* (M) and *Queue time* (Q). *Service time* is the amount of time that the Web service takes to perform its task. *Message delay time* is determined by the size of the message being transmitted/returned and the load on the network through which the message is being sent. *Queue time* is the delay caused by the load on the system where the Web service is deployed. The *Total invocation time* (T) for Web service *s* is given in the formula below:

$$T(s) = S(s) + M(s) + Q(s).$$

Our enactment system uses the centralized approach and is therefore controller based, allowing us to do performance evaluation by instrumentation on the controller side. We determined the total invocation time for each Web service and then calculated the time for each of the components that make up total invocation time. Message delay time was calculated by invoking a ping function for each Web service. XML messages were sent and received, but the Web service performed no work. Service time was calculated by running tests against the Web service in an environment where the load and queuing delay for the service were controlled. The queue time was determined by running the test in an environment where the Web service was loaded with requests. Figure 3 shows the average time taken by the primary Web services in our test enactment.

Performance evaluation can help in adapting the Web process based on the Quality of Service requirements. Figure 3 shows the distribution of the overall time for one of our tests. In Figure 3, the *ReleaseOrder* Web service has a high queuing time. This may indicate that the system hosting the service is not able to handle the load. Replacing the *ReleaseOrder* Web service with another service may improve the quality of service for the Web process.

The performance evaluation of Web services needs to be done in a controlled manner. This requires that one be able to control the load on the system during testing. The distribution and autonomy of Web services makes meeting this requirement difficult. Using simulation to predict the performance of a Web process is therefore very useful.

6 SIMULATION

Simulation helps in determining how composed Web services will perform when they are deployed and may also uncover structural errors in the design. Simulation serves as an important step in designing efficient processes. The WPDT process designer is integrated with the JSIM simulation system. WPDT is linked to a JSIM Model Generator to generate JSIM simulation model from the WSFL process model. Currently JSIM simulates the controller based enactment of Web processes (see Figure 4). Simulation of distributed enactment is underway. As both WSFL process

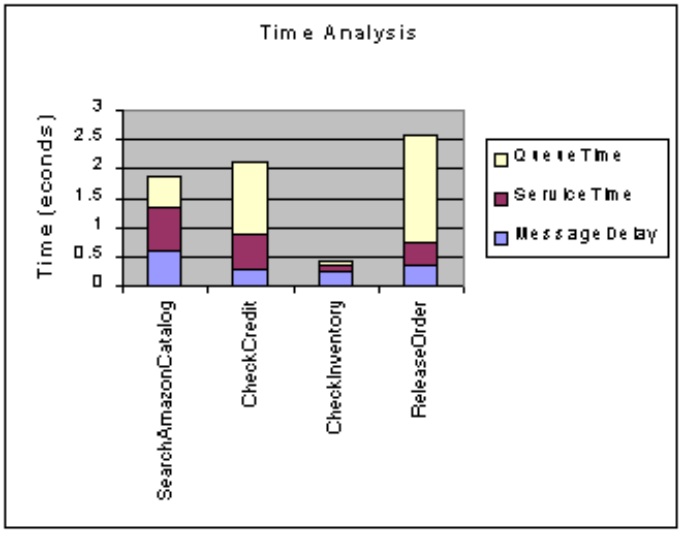

Figure 3: Timing Results for the Book Purchasing Web Process

model and JSIM simulation model are represented as digraphs, mapping from the WSFL model to the JSIM Model is straightforward. Control links in the process model map to transports in the JSIM simulation model. Activities in the process model map to facilities in the simulation model. Thinking for a moment in the other direction, one could use WFSL like specifications as a basis for developing XML standards for specifying simulation models.

6.1 JSIM Simulation

The latest version of JSIM, a Java-based simulation and animation environment (Nair *et al.* 1996; Miller *et al.* 1997), contains several features that support the simulation of Web processes. JSIM simulation models are constructed using the following basic components:

- Source Nodes: Generate entities using an inter-arrival time produced by a random variate.
- Server Nodes: Provide service to entities using a service time produced by a random variate. Servers may have one or several service units.
- Facility Nodes: Behave like a server node but also provide a queue for waiting entities.
- Sink Nodes: Consume entities and capture statistical information about the entities.
- Transports: Edges that connects two nodes.
- SimObjects: Instances of simulation entities.

Messages sent to or received from Web services are modeled as *SimObjects*. The Web services in the process are modeled as *Facility* or *Server* nodes, and the communication channels between Web services are modeled as *Transports*.

The primary enhancements, conditional routing and AND splits, to support the simulation of Web services are discussed briefly.

6.1.1 Conditional Routing

In prior versions of JSIM, an entity leaving a node would probabilistically choose an out edge to transport it to the next node. JSIM has been enhanced to allow an out edge to be selected based on the values of simulation entity attributes. The model developer is given the option to add additional attributes to instances of the *SimObject* class and out edge selection criteria to instances of the *Transport* class. The selection criteria are specified as Java condition expressions.

AND Splits allow an entity to choose more than one out edge to transport it. When an entity encounters an *AND Split*, the entity is cloned and each copy of the entity exits the node on a different out edge. Each of the copies of the entity continues to traverse the graph until they encounter a node containing an *AND Join*. Each copy of the entity will wait at this node until all other copies arrive. Once all of the copies arrive, they are joined by rule-based merging of the attribute values of the copy entity with the attribute values of the original entity and placing results into the attributes of the original entity. The copies are then removed from the model, and the original entity may continue to traverse the graph.

The time for the original entity to be transported from the node with the *AND Split* to the node with the *AND Join* is considered to be the greatest amount of time that it takes any of the copies to be transported between the nodes.

When simulating Web processes, the JSIM model takes as input the distribution functions characterizing the Web services. The service time distribution functions of the Web

services, are used to generate service times for characterizing the facility/server nodes in the JSIM model specification. These distribution functions can be computed by performance evaluation tests as explained earlier or obtained from the service providers.

7 BUILDING SIMULATION ENVIRONMENTS OUT OF WEB SERVICES

Currently, the World Wide Web is mainly a collection of documents that are searchable via keywords. Enormous efforts are currently underway to transform it into a more effective Web. Simultaneously, Web documents are being made more meaningful and functional capabilities are being added. These efforts are referred to as the Semantic Web and Web Services, respectively. The intent of the semantic Web is to allow users to find more of the information they want and less of the information they do not want. In other words, hits will be better targeted. With the incredible aggregate computing power and newer, higher network bandwidths, it only makes sense to provide services on the Web. This has been done for awhile in a proprietary fashion (*e.g.*, ordering a book from Amazon.com). For the first time in the history of computing, infrastructure is being developed to provide services in a standardized and interoperable fashion, on a global scale.

7.1 Types of Web Services

Many types of Web services can be useful in simulation. We discuss three types below.

7.1.1 Whole Models as Web Services

Perhaps the most useful is to make complete simulation models available as Web services. Certain sites develop an expertise with certain types of simulation (*e.g.*, a highway traffic simulation or weather simulation) and if one wishes to perform such a simulation, they could simply send a SOAP message to the appropriate site which characterizes the scenario they wish to study. The model Web service could charge on a per use basis or lease out its service.

7.1.2 Environmental Components as Web Services

Drilling down a bit, a simulation may use several major components such as Databases, Spreadsheets, Knowledge Bases, Visualization Tools, OLAP Tools, Data Mining Tools, Scenario Managers, Optimizers and Animators. Components with infrequent interation are the best candidates to be separated out as Web services. The service may be provided by a third party, but might well be provided within the same organization. The Web service paradigm would be followed for the purposes of standardization, interoperability, maintainability and flexibility (*e.g.*, if the company's database is replaced, so long as the relevant Web service is redeveloped, none of the simulation models using the database need to be changed).

- The first three components/resources (Databases, Spreadsheets and Knowledge Bases) are mainly data/information/knowledge sources and sinks which would be primarily used at the beginning and end of a simulation and hence are excellent candidates for becomming Web services.
- The second three components/tools (Visualization Tools, OLAP Tools and Data Mining Tools) can be decoupled from the simulation and simply access an information resource, so they are also well suited.
- The final three components (Scenario Managers, Optimizers and Animators) are more tightly coupled with simulation, but are still separable enough to form cooperative Web services. Depending on its form an animator may be either tightly-coupled (*e.g.*, has access to the simulation state) or loosely-coupled (*e.g.*, has access to a trace that is say stored in a database).

The beauty of this approach is that one could develop a simulation or general-purpose simulation engine and simply link it with other state-of-the-art components. Some of these components such as databases and visualization tools are better left to other communities/industries rather than having to be redeveloped by the simulation community. As Web services, they would then be readily and easily available.

7.1.3 Model Federates as Web Services

The final step in utilizing Web service technology would be to code model federates as Web services. If the interaction rate between federates is not too high, the disadvantage of increased overhead may be outweighed by the interoperability benefits. In cases in which enforcement of cauasality is not important, this change would be straightfoward. If causality is to be enforced, then new infrastructure needs to be developed. Already, there are efforts to provide composite Web services with transactional capabilities (Mikalsen *et al.* 2002). Provision for casuality could similarly be provided.

Such an effort would parallel the work done on the High-Level Architecture (Frederick *et al.* 2000) and might even serve as a spark to ignite commercial efforts at standardization for distributed/federated simulation, the practicality of which has so far been limited to the military sector.

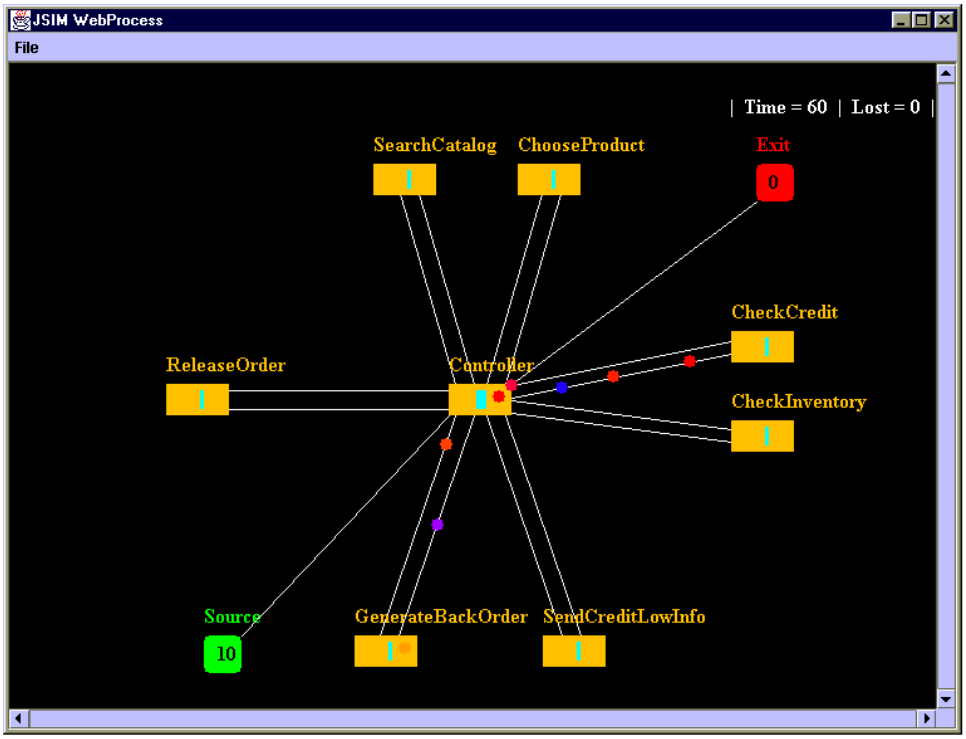

Figure 4: JSIM Model for the Book Purchasing Web Process

7.2 Evolution of JSIM

During the past decade researchers have explored the use of component-based software to develop modular simulation environments. These environments allow developers to treat simulation models and other data resources as components that can be assembled to create more elaborate simulation models. The JSIM simulation environment currently uses Java Beans technology to link the simulation models (Miller *et al.* 1998). JSIM also uses XML-based messaging to address interoperability issues (Huang and Miller 2001). Other researchers have used an agent-based environment for linking simulation models, data resources and other components (Wilson *et al.* 2001; Mills-Tettey *et al.* 2002).

The JSIM project is currently in the process of conversion to Web service technology. Services will be composed both statically and dynamically for simulation applications. Services (models, resources, tools or federates) may be found on the Web using registries/repositories. Decriptions in the form of WSDL, WFSL, DAML-S or some newer description/information modeling scheme will be used. Ontologies will provide a semantic basis for the terms used in these descriptions providing greater pricision in finding the appropriate Web service. Web service technology is being used to make JSIM models available as Web services. We are in the process of enhancing JSIM so that simulation data can be stored and retrieved in databases using Web services. JSIM currently has the ability to represent simulation data as XML messages which can be stored in a database (Huang and Miller 2001).

We are currently working on JSIM to enhance the implementation (Huang and Miller 2001) to make use of Web services in the composition of JSIM model federations. Each component in a federation may be a Web service. The model federates (JSIM models) will be Web services as will the model agents that control the execution of single models, the database agents that act as interfaces for database access and the scenario agent that controls the execution of the overall federation. These components will communicate with one another using SOAP messages.

8 CONCLUSIONS AND FUTURE WORK

The paradigm of Web services is being promoted and standards are being worked out to ease the adoption of Web services. This paper discusses the problem of composing Web services to get a better performing process. We present performance analysis and simulation as tools that can aid a user in composing Web processes. The Web Services Designer Tool (WSDT), facilitates the composition of Web services. This designer is packaged with the JSIM simulator allowing users to compose a process and, if the service time distributions for the activities in the process are known, simulate the process on the fly. Using JSIM users can do

"what-if" senarios and visualize the Web process in action before enactment.

From the other direction, JSIM is being converted to use Web service technology. This is facilitated by the fact that JSIM already supports XML messaging.

REFERENCES

Ankolekar A., M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, S.A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. 2001. DAML-S: Semantic markup for Web services. In *Proc. of the International Semantic Web Working Symposium, Stanford, CA.*

Benatallah B., M. Dumas, Q. Sheng, and A. Ngu. 2002. Declarative composition and peer- to-peer provisioning of dynamic Web services. In *Proc. of the International IEEE Conference on Data Engineering*, San Jose, USA,.

Cardoso J. and A. Sheth. 2002. Semantic e-workflow composition. In *Journal of Intelligent Information Systems* (submitted).

Cardoso J., A. Sheth, and J. Miller. 2002. Workflow quality of service. In *International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference*, Valencia, Spain, Kluwer Publishers.

Cardoso J., J. Miller, A. Sheth, and J. Arnold. 2002. Modeling quality of Service for workflows and Web service processes. In *The VLDB Journal* (submitted).

Christensen E., F. Curbera, G. Meredith, and S. Weerawarana. 2001. Web services description language (WSDL) 1.1. Available online via <http://www.w3.org/TR/wsdl> [accessed August 19, 2002].

Florescu D., A. Grunhagen, and D. Kossman. 2002. A XL: An XML programming language for Web service specification and composition. In *Proc. of the Eleventh International World Wide Web conference*, Honolulu, HI. Available online via <http://xl.in.tum.de/publ/www2002.html> [accessed August 19, 2002].

Frederick K., R. Weathery, and J. Dahmann. 2000. *Creating computer simulation systems.* New Jersey: Prentice Hall.

Huang X. and J. Miller. 2001. Building a Web-based federation simulation system with JINI and XML. In *Proc. of the 34th Annual Simulation Symposium*, 145-150. Seattle, WA.

Kochut K., A. Sheth, and J. Miller. 1999. Optimizing workflow. In *Component Strategies*, 1(9): 45-57.

Leymann F. 2001. Web service flow language (WSFL) 1.0. Available online via <http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf> [accessed August 19, 2002].

Mikalsen T., S. Tai, and I. Rouvellou. 2002. Transactional attitudes: Reliable composition of autonomous Web services. Available online via <http://xml.coverpages.org/ni2002-04-03-a.html> [accessed August 19, 2002].

Miller, J., J. Cardoso, and G. Silver. 2002. Using simulation to facilitate effective workflow Adaptation. In *Proc. of 35th Annual Simulation Symposium*, 177-181. San Diego, CA.

Miller J., Y. Ge, and J. Tao. 1998. Component-based simulation environments: JSIM as a case study using Java Beans. In *Proc. of the 1998 Winter Simulation Conference*, 373-381. Washington, DC.

Miller J., R. Nair, Z. Zhang, and H. Zhao. 1997. JSIM: A Java-based simulation and animation environment. In *Proc. of the 30th Annual Simulation Symposium*, 31-42. Atlanta, GA.

Miller J., D. Palaniswami, A. Sheth, K. Kochut, and H. Singh. 1998. WebWork: METEOR's Web-based workflow management system. In *Journal of the Intelligent Information Management Systems*, 10(2): 185-215.

Miller J., A. Sheth, K. Kochut, X. Wang, and A. Murugan. 1995. Simulation modeling within workflow technology In *Proc. of the 1995 Winter Simulation Conference*, 612-619. Arlington, Virginia.

Mills-Tettey A., G. Johnston, L. Wilson, J. Kimpel, and B. Xie. 2002. The Abels system: Designing and adaptable interface for linking simulations. In *Proc. of the 2002 Winter Simulation Conference*, San Diego, CA.

Nair R., J. Miller, and Z. Zhang. 1996. JSIM: A Java-based query driven simulation environment. In *Proc. of the 1996 Winter Simulation Conference*, 786-793. Coronado CA.

Narayanan S. and S. Mcllraith. 2002. Simulation, verification and automated composition of Web services. In *Proc. of the Eleventh International World Wide Web Conference*, Honolulu, HI.

Sheth, A., K. Kochut, J. Miller, D. Worah, S. Das, C. Lin, Lynch J., Palaniswami, D., and I. Shevchenko. 1996. Supporting state-wide immunization tracking using multi-paradigm workflow technology. In *Proc. of the 22nd International Conference on Very Large Databases*, 263-273. Bombay, India.

Sipani S., K. Verma, S. Chandrasekaran, X. Zeng., J. Zhu, D. Che, and K. Wong. 2002. Designing an XML database engine: API and performance. In *Proc. of the 40th Annual Southeast ACM Conference*, 239-245. Raleigh, NC.

Stormer H. 2001. Task scheduling in agent-based workflows. In *Proc. of the International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce*, Wollongong, Australia.

Thatte S. 2001. XLANG: Web services for business process design, Available online via

`<http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm>` [accessed August 19, 2002].

Wilson, L., A. Burroughs, A. Jumar, and J. Sucharitaves. 2001. A famework for linking distributed simulations using software agents. In *Proc. of the High Performance Computing Symposium*, 147-154. San Diego, CA.

AUTHOR BIOGRAPHIES

SENTHILANAND CHANDRASEKARAN is an MS student in Computer Science at the University of Georgia. His research interests include Distributed Computing, Web services, Web process composition and simulation.

GREGORY SILVER is a PhD student in Computer Science at the University of Georgia. His research interests include simulation and distributed systems.

JOHN A. MILLER is a Professor of Computer Science at the University of Georgia and is also the Graduate Coordinator for the department. His research interests include database systems, simulation and workflow as well as parallel and distributed systems. Dr. Miller received the B.S. degree in Applied Mathematics from Northwestern University in 1980 and the M.S. and Ph.D. in Information and Computer Science from the Georgia Institute of Technology in 1982 and 1986, respectively. During his undergraduate education, he worked as a programmer at the Princeton Plasma Physics Laboratory. Dr. Miller is the author of over 70 technical papers in the areas of database, simulation and workflow. He has been active in the organizational structures of research conferences in all three areas. He is an Associate Editor for ACM Transactions on Modeling and Computer Simulation and IEEE Transactions on Systems, Man and Cybernetics as well as a Guest Editor for the International Journal in Computer Simulation and IEEE Potentials.

JORGE CARDOSO received a B.A. (1995) and a M.S. (1998) in Computer Science from the University of Coimbra (Portugal). He received his PhD also in Computer Science from the University of Georgia (USA, August 2002). His PhD degree is in the area of workflow management. His research work concentrated on workflow QoS management and semantic composition of workflows.

AMIT P. SHETH is a Professor of Computer Science and the director of the Large Scale Distributed Information Systems (LSDIS) Lab, at the University of Georgia. His research interests include semantic interoperability, semantic Web and global information systems with applications to digital libraries, video and digital media applications utilizing broadband, and e-commerce, and enterprise integration with emphasis on multi-organizational business processes, workflow coordination and collaboration technologies. Before joining UGA in 1994, he served in Research and Development groups at Bellcore (now Telcordia Technologies), Unisys, and Honeywell. He has founded two high-tech companies. His research has led to several commercial products and applications. He has published over 125 papers and articles, given over 100 invited talks and colloquia, (co)-organized/chaired six conferences/workshops, and served on over 60 program committees.