ANALYSIS AND COMPOSITION OF DISCRETE EVENT ORIENTED SIMULATIONS USING DISTRIBUTED WEB TECHNOLOGIES

Aman Atri [1], **Felix Breitenecker** [1], **Katharina Breitenecker** [1]

[1]Vienna University of Technology, Institute for Analysis and Scientific Computing
Wiedner Hauptstrasse 8-10, 1040 Vienna , Austria

aman.atri@tuwien.ac.at(Aman Atri)

Abstract

The rising facilities of higher level protocols for internet communication have provided successful implementations in current web development. Web applications are not merely meant to display information and content, but work interactively using standardised conventions. These technology is commonly used in many social platforms where different kind of applications interact together regardless of the actual programming language they were written in.
The purpose of this paper is to use these communicaton layers for a platform independent web based e-learning system for discrete event oriented simulation. The software shold be used for academic teaching purposes. The server-sided simulation is basically a multi-tier software architecture. The client communicates only through the W3C defined standards such as XML,HTML, Javascript and JSON. Thus the client's browser does not require any third party software like Flash, Java Plugin, etc. This convenience allows the simulation environment to swap the backend simulation engines so that the same experiment can be executated with different simulators for e-learning portals.

Keywords: Discrete Event Simulation, Web-Based Simulation, AJAX Webinterfaces, E-Learning

Presenting Author's Biography

Aman Atri studied Software and Information Engineering at the Vienna University of Technology. His bachelor thesis analyses automated proofs for model checking in temporal logic. After his bachelors program he pursued with the master course Software Engineering and Internet Computing. His master thesis deals with discrete simulation and visualisation schemes for the web. This work has been dilated in his PhD thesis where he is analysing and developing service oriented simulation frameworks and interoperable connectivity of different simulators. He is working as a research assistant at the Vienna University of Technology (Institute for Analysis and Scientific Computing).

1 Introduction

Internet based applications have reached a state where the local bounding to one certain webserver is no more required. Applications have now become loosely coupled components which communicate in a distributed way. Modern web frameworks use Web 2.0 technology which can be found in any common social platform. The idea of this work is to reuse these methods to develop a a communication mechanism for distributed event-oriented discrete simulation using standardised protocols and a large scalable multi-tier architecture.

While simulators are becoming more and more complex the goal is to keep the controlling panel as simple as possible. Where a few years ago the basic functionaliy of the web was to display content which was formated in HTML/Javascript/CSS on the clients browser it has now become a interactive part where not only the display of data and information is the main task but the bidirectional manipulation and computation of the provided data. In the field of discrete simulation web based simualtors executed the simulation on the local client using a software plugin such as Java Applets, Flash, Active-X, etc. These concept is usually restricted due to security reasons like memory restrictions, access to data and locally stored files. Latest web standards allow us now to use local and remote resources through transparent network protocols[1]

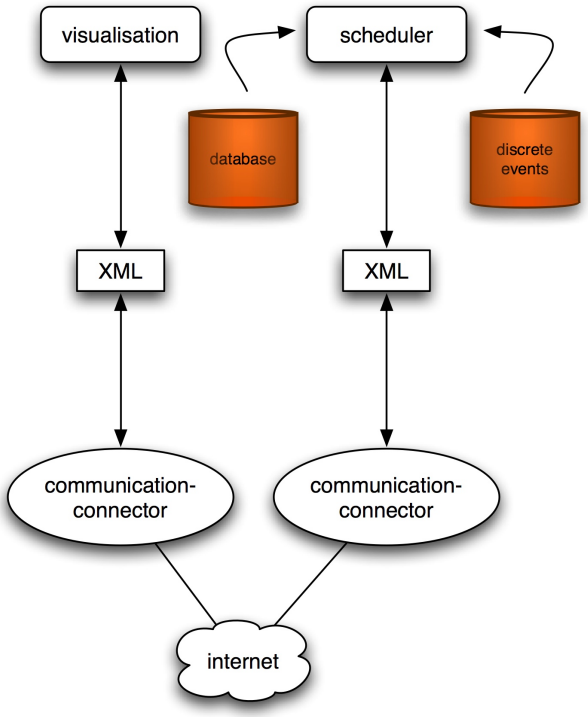

Fig. 1 Communication Channels

2 Software Architecture

In terms of distributed computing we have certain kind of boundries which can be optimised for a flexible performance. In discrete event oriented simulation a lot of complex datastructures have to be passed over the network for remote computation. These data has then to be sent to other nodes for further proceeding like validation, visualisation [1], etc and finally transferred back to the client for the actual simulation experiment. These complex communication needs to be implemented in a bidirectional way.

2.1 Bidirectional Communication

Fig.1 shows that every server node is communicating to a translator which understands both incoming and outgoing commands. As for XML is an open standard and a lot of message passing protocols have already been implemented in XML the communicators use this language also for exchanging the data.

2.2 Service-Oriented Simulation

Execution of remote methods on the server side are represented as so called web services. A web service is introduced by defining a Web Service Description Language (WSDL) which is also formated in XML. A WSDL file describes which remote methods are offered to the clients and what kind of parameters and preconditions are required. For the simulating client the data is transferred in an object-oriented fashion while on the lower level these objects are marshalled into XML-Objects and then by-passed trough the HTTP protocol.[2, 3, 4]

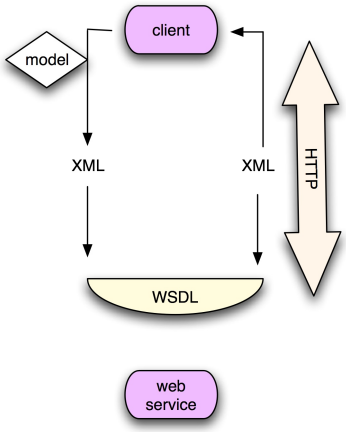

Fig. 2 Service-Oriented Architecture

As for the HTTP protocol is designed to be a stateless protocol the web server looses the information of the client which is requesting a resource after the transmission has been completed successfully. In order to keep in touch with the web service a multi-tier architecture is applied to the system. Thus the client doesn't directly connect to the web service but to a component in between which is implemented as a Java Servlet. A positive side effect is the load balancing of the server machine. Now that in case the web service is overloaded and cannot proceed further until the previous tasks have

been completed, the servlet can dispatch any future requests to another web service which is currently idle.

2.3 Sequential Simulation vs. Parrrallelism

Different web services can be switched together or can work even concurrently. Building simulation networks where unpredictable number of components work together or may drop out implies to design an architecture which implements the following criteria[5]:

- The client side should be simple and contain only a few classes to decrease overhead traffic
- Discrete events triggered remotely have to be recognized and verified in case of data is lost or falsified during data transfer
- Fault tolerance mechanisms have to grant the exchangeability of components in case of loss of connectivity. The simulation engine has to be notified when some remote modules are not reachable

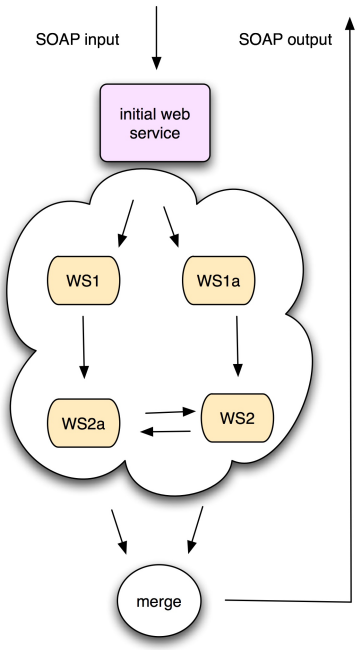

Fig. 3 Webservices as Pipes

Now that the web services reside on different physical machines the task can be distributed accordingly. A machine with a fast CPU or GPU might be engaged computing complex numerical tasks while a large capacity with internal backup system can be utilised as a host which feeds and fetches data from a database management system (DBMS). Fig. 3 illustrates this work flow allocation. [6]

3 Persistence of discrete Models

In discrete simulation the data is not only provided by a randomisation based on a statistical distribution but often depends on already provided data which comes in a relational database. As our models are designed and implemented in an object oriented fashion and direct giving access to the clients might be risky concerning security policies the application server has the ability to convert relational database tuples into objects and vice versa. As described in Fig. 4 the model is converted into XML using hibernation and then stored in a relational database. Thus the client doesn't need do deal with the details which database driver is used or how the tables look like. This information abstraction allows a hig-level scalability as for the database can be extended and modified without chaning the code on the client side.

All these features (web services, SOAP, web server, hibernation) are included in the Java2 Enterprise Edition (J2EE) specification API. From the client's point of view it is a single host which executes the requested commands but in the backend it is a whole network which can optimize the workflow dynamically.[7]

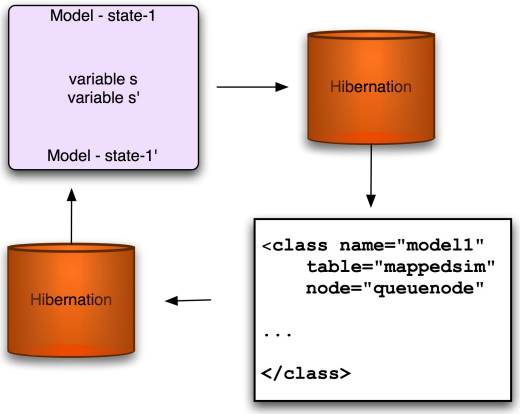

Fig. 4 Persistance with Hibernate

Even dispatching to external applications is done in a hidden way. So the user can for instance execute MATLAB code without even having MATLAB installed locally. The MATLAB Compiler for Java ™can compile MATLAB code into Java classes which can be nested in the repository of the application server.

4 Web-Based Visualisation

The server sideded pattern is a complex combination of different components and modules which work independently from the client software. As for most of the common modern programming languages provide bindings and wrappers for SOAP and web services even the server application modules can be written in different programming languages. The goal is to separate the actual simulation language from the compiled and executed programming language. The user should be able to start, control, manipulate and parameterise the simulation experiment with a minimum of overhead. The web application is designed to be used for e-learning purposes in education, rather than heavy-loaded perfor-

mance in the browser.[8]

4.1 XML and Javascript

Now that HTTP is a stateless protocol but the communication between the browser and the web server should be bidirectional the data which is being transferred is converted into XML queries and posted in an asynchronous manner. The visualisation at the client sided display is done via Javascript. This paradigm is widely used on many social platforms commonly known as Web 2.0 technology. The Asynchronous Javascript And XML (AJAX) code includes several features like:

- JavaScript Object Notation (JSON): As in discrete event oriented simulation a lot of complex objects are processed concurrently, XML might lead to a traffic and parsing overhead. JSON provides a thin and simple and easy to parse syntax where the nested paramenters can be restored as they were sent through the browser.
- XML-RPC: Remote execution of methods where all functions and their parameters are converted in to an XML description.
- Dynamic source execution without reloading the page: The user doesn't need to reload the page after a parameter as been posted. This feature is quite handy for fluent animations. The AJAX connector for Matlab cross compiled Java classes even support dynamic rendering of Matlab figures. The user doesn't need to reload static (but dynamically generated) images but can visualise the experiment directly by manipulating and passing parameters. [9]

The interactive web based simulation enables the user to enter parameters or modify them using the browser. Usually if the application is not running on on the client side the browser sends this parameter to the web server using the HTTP protocol. The request is matched with the unique session of the user and the session values are updated with the latest parameters The simulation web application sends the result to browser and the page is reloaded. Because of this latency during the HTTP requests the visualisation cannot run fluently. Ajax technology is a work around to avoid networking overhead.

The main focus is not only to optimize single algorithms but the whole system. Event oriented systems dont interact trough a stream of information and data but with synchronized events and require an architecture which is mostly suitable if we use the web as a global computation platform for a frictionless integrity of storage and computing units.[5]

4.2 Modeling using GWT

Generic discrete simulators with graphical interfaces can now be designed easily to work also as modelling and visualisation application. The user can drag and drop components like random number generators, waiting queues, sinks, loops, timers, etc. with just a few

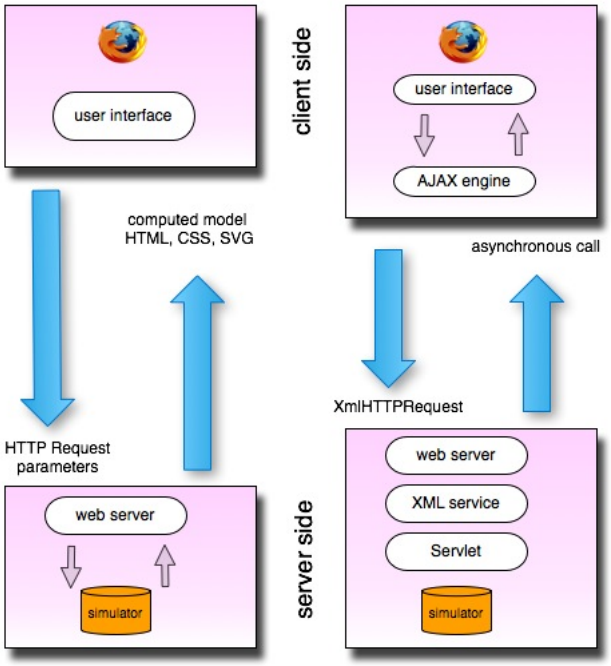

Fig. 5 AJAX and HTTP

mouse clicks and can justify the parameterisation. Usually this graphical inputs have to be uploded to the web server through forms or applets. Now with AJAX doing the synchronsiation in the background the objects can be created remotely and any state changes on the client side will be updated with the information given from the XML webserver.

The simulating web server has to deal with lots of different kind of browsers and not every browser acts according to the given JavaScript code. Sometimes there might be inconsistent issues with the graphical display. With the help of GWT (Google Web Toolkit) it is now easy to design a web page layout and asynchronous algorithms. The actual code is written in the Java programming language using the GWT framework. The code is then compiled into Javascript and XML code and supports most of the modern common browsers. GWT even supports the Java Servlet API and is capable to generate dynamic code while beeing executed in a Servlet Container like Apache Tomcat. Even XML-RPC and JSON code can be easily debugged because errors are not overridden by the browser but the actual Java compiler will throw an error. As a result the output code is compatible with most browsers. [1]

The data can be converted from Java objects to JSON objects which on the other hand can be transferred to another domain. This cross-domain feature allows the web services to reuse the objects without requesting them from the client again.

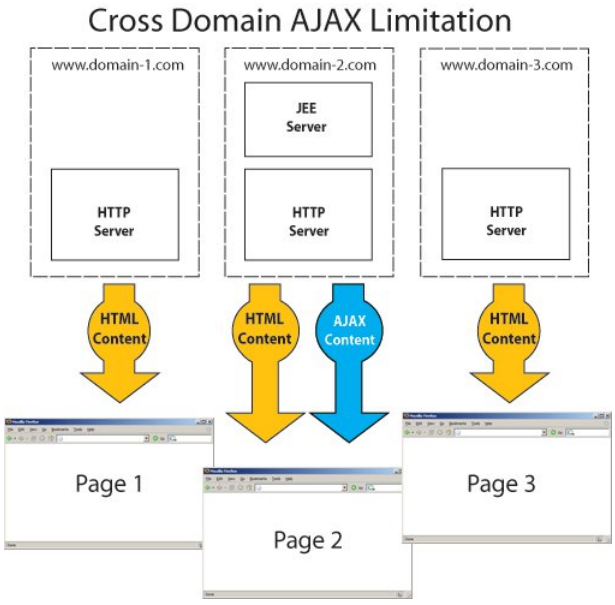

Fig. 6 Cross domain references

5 Simulating in Tuple-Spaces

In large scale networks there is a high risk, that certain kind of bottleneck effects can occur. This happens usually if one process is in starvation mode or while other processes might be hung up in a deadlock. The client cannot distinguish between:

- Overload: The server sided application is requesting for more resources
- Latency: The web service is still waiting for results from other threads. This can often lead to timeouts.
- Crash: Any or many modules have been crashed and cannot recover the tasks automatically
- Data loss: During the transmission of data some information might have been lost so that further proceeding is not in a correct order.
- Deadlock: Different services want to have exclusive rights one resource.
- Network problem: There might be some problems in the network which leads to latency.

To avoid such situations the simulation plattform and the server nodes are surrounded by a global multiple persistance middle-ware. This so-called *tuple-spaces* provide scalability, resource sharing, and fault tolerance. A tuple space provides an unified memory (altough the resources reside on different physical machines) with the following features:

- Every client gets an update of the state of all entities at any time requested.

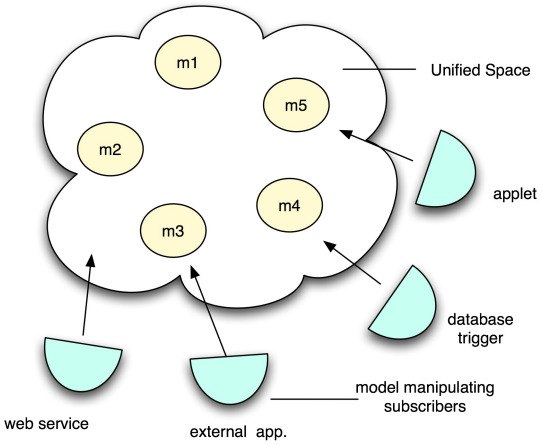

Fig. 7 Multiple presistence (unified space)

- Any manipulation of the state of a model has to be transactional. That means every client has to be notified that an update has been committed. If a client does not respond positively than either the transaction has to be withdrawn or the corresponding client has to be listed as non-active.
- Every entity which is accessible to all clients has at least one copy of itself in the system.

Thus a tuple space can provide duplicated resources and update the originial entities. A reference implementation of a tuple space is the open source project MozartSpaces[10].

6 Conclusion

The usage of web technologies and communication protocol is to provide a transparent and easy to scale system to provide a web based simulation environment for discrete event oriented simulation in education. The web application is designed to be used in e-learing and education for academic courses regarding simulation.

The software design uses XML as a description language for building the models and for communication with the application server. The user is not aware that the simulation environment uses different internal services for optimizing the workflow of the experiment. Different programming languages can be interconnected and different webservices can share objects in a global space. To provide fault tolerance and stability for the whole system a global backup system (called tuple space) is ensuring that data won't get lost and ev-

ery resource available through at least one duplicate instance.

7 References

[1] N. Nagele A. Atri. Web-based discrete simulation services for education in modelling and simulation. In F. Breitenecker I. Troch, editor, *Proceedings MATHMOD 09 Vienna - Full Papers CD Volume*, volume 35. ARGESIM / ASIM Vienna, 2009.

[2] Jerzy Tyszer. *Object-Oriented Computer Simulation of Discrete-Event Systems*. Springer, 1st edition, 5 1999.

[3] Bernd Page and Wolfgang Kreutzer. *The Java Simulation Handbook: Simulating Discrete Event Systems with UML and Java (Berichte Aus Der Informatik)*. Shaker Verlag GmbH, Germany, 11 2005.

[4] M. Gyimesi. *Simulation Service Providing unter Verwendung von Web Service Technolgie*. PhD thesis, Vienna University of Technology, 2005.

[5] A. Atri F.Breitenecker N.Nagele S.Tauboeck. Distributed discrete simulation on the web. *Proc. 20th European Modeling and Simulation Symposium EMSS2008*, pages 392–397, 2008.

[6] Schahram Dustdar, Harald Gall, and Manfred Hauswirth. *Software-Architekturen fr Verteilte Systeme: Prinzipien, Bausteine und Standardarchitekturen fr moderne Software (German Edition)*. Springer, 1 edition, 7 2003.

[7] Akmal Chaudhri, Mario Jeckle, Erhard Rahm, and Rainer Unland, editors. *Web, Web-Services, and Database Systems: NODe 2002 Web and Database-Related Workshops, Erfurt, Germany, October 7-10, 2002, Revised Papers (Lecture Notes in Computer Science)*. Springer, 1 edition, 4 2003.

[8] Eben Hewitt. *Java Soa Cookbook*. O'Reilly Media, 1 edition, 3 2009.

[9] Nicholas C. Zakas, Jeremy McPeak, and Joe Fawcett. *Professional Ajax, 2nd Edition (Programmer to Programmer)*. Wrox, 2 edition, 3 2007.

[10] MozartSpaces A tuple space implementation. http://www.mozartspaces.org.

[11] Richard A. Kilgore, Kevin J. Healy, and George B. Kleindorfer. The future of java-based simulation. In *WSC '98: Proceedings of the 30th conference on Winter simulation*, pages 1707–1712, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[12] Jasna Kuljis and Ray J. Paul. A review of web based simulation: whither we wander? In *WSC '00: Proceedings of the 32nd conference on Winter simulation*, pages 1872–1881, San Diego, CA, USA, 2000. Society for Computer Simulation International.

[13] Charles Marr, Christopher Storey, William E. Biles, and Jack P. C. Kleijnen. A java-based simulation manager for web-based simulation. In *WSC '00: Proceedings of the 32nd conference on Winter simulation*, pages 1815–1822, San Diego, CA, USA, 2000. Society for Computer Simulation International.

[14] Robert E. Shannon. Introduction to simulation. In *WSC '92: Proceedings of the 24th conference on Winter simulation*, pages 65–73, New York, NY, USA, 1992. ACM Press.

[15] Matlab Java Compiler (Mathworks). http://www.mathworks.com/products/compiler/.

[16] Javascript Object Notation (JSON). http://www.json.org.

[17] Java Enterprise Edition Glassfish. http://java.sun.com.

[18] SOAP Specification W3C. http://www.w3.org/tr/soap.