WEB-BASED DISCRETE SIMULATION SERVICES FOR EDUCATION IN MODELLING AND SIMULATION

A. Atri[1], N. Nagele[1], N. Popper[2]
[1]Vienna University of Technology, Austria, [2]DWH - Simulation Services, Austria

Corresponding author: A. Atri, Vienna University of Technology, Inst. f. Analysis and Scientific Computing
1040 Vienna, Wiedner Hauptstraße 8-10, Austria, atri@asc.tuwien.ac.at

Abstract. The swift development of web applications has undergone major changes in recent years. The focus is no longer on just mere document delivery over the network but more on the dynamic interaction of the human resource using the browser and the web application which publishes the content. Manipulation is not just the task of the webmaster but more of the community which corresponds with the web application. This paradigm of the *new* web is commonly mentioned as Web 2.0. The goal of this work is to analyse the technological background of Web 2.0 applications and utilise AJAX components for web-based discrete event oriented simulators (DES) for e-Learning environments used for academic lectures. Furthermore the architectural design for distributed web simulation and their interactivity with the graphical frontend in the browser are discussed.

1 Motivation

The modern web is continuously adapting new web technologies to provide appropriate and easy to use services. Whereas a few years ago the basic functionality of the web changed from being a mere content display to an interactive content manipulation system with separeted handling of data and its actual representation, its now the generic encapsulation of the information which makes the web usable for more purposes than illustration. It is not only important that the content and its formatting rules are distinguished from each other, but also that the extracted data of the content can be passed on to external applications for a smooth postprocessing. In the field of discrete event oriented simulation, web-based simulators are commonly designed for a local execution within a browser through a plugin-based environment such as Flash or Java Applets. These simulators are often bound the local restrictions of the client ambit such as browser security policies, local memory, version incompatibilities of the plugin software, etc. Latest web standards allow us to use the local and remote resources through transparent networking protocols. Data sharing within open standards allow a more decent interactivity between different simulators and application services. This work includes strategies how this data-bundling is performed and optimized.

2 Modelling and deployment over the web

When it comes to web-based simulation the question rises what would be the most suitable way to transport the data. Network-based simulation have usually an optimisation protocol based upon other layers like TCP/IP etc, and often operate on a proprietary API. The goal of any network based solution for simulation services is to keep the size of the data as small as possible with respect to a high transfer rate. Generic protocols like the Hypertext Transfer Protocol (HTTP) bound the simulator to limitations because of their stateless nature. The traditional unidirectional way is a simple socket query with POST and GET messages, where the stream is closed once the data has been shipped. To remember the clients identification unique IDs (Session IDs) or Browser Cookies are used for *h*and-shake-communication. The data is passed as strings over a HTML form and the whole page has to be reloaded as the web server processes the provided information. This latency cannot be pre-computed and hence the whole process becomes blocking. MATLAB has also introduced a method where a Simulink model can be deployed over the web using server-sided cgi-bin and the Real-Time Workshop toolbox. The procedure is actually difficult to automatise as for the conversion of the simulink model standalone executable for the web has to be done manually via MATLAB and the MATLAB compiler. The output is shown as as scalable vector graphic images (SVG) and then sent via an HTML interface to the end user.

2.1 Asynchronous Communication

Due to the behaviour of classical stateless protocols like HTTP data transfer is restricted in direction, chronological order and as a negative side effect it produces a traffic overhead creating potential falsified discrete triggering data. This might result in latency effects in visualization on the client side. There are higher-level approaches of web programming frameworks to implement visualisation trough:

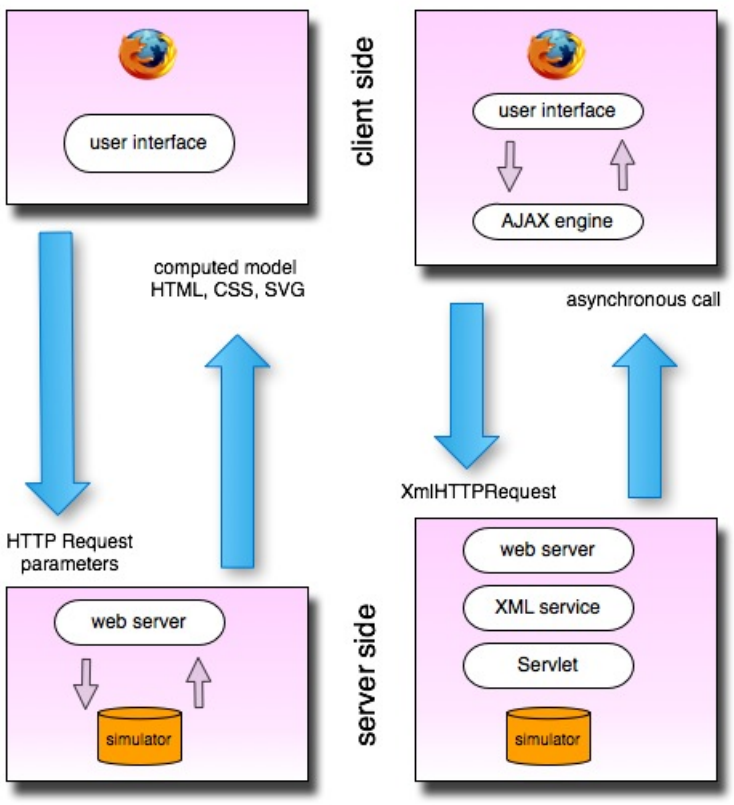

Figure 1: Message passing with traditional HTTP and AJAX queries.

- Dynamic code generation with asynchronous data delivery from the server
- A statically pre-compiled graphical application which can directly access the service and might be able to overtake some portions of the computing unit

The first approach is implemented in Web 2.0 fashion using XML mapping between clients and server. Ajax (Asynchronous JavaScript and XML) emulates the characteristics of a stateless protocol. In case of required data update, the application is able to receive the new visualisation information in background without interrupting the actual simulation process.[1]

Figure 1 shows the difference between the traditional HTTP requests and the new overlay using an AJAX engine. The key component for this framework is the API for the so-called *XMLHttpRequest* object which delivers the interface for asynchronous communication with the XML web server [10]. With XMLHttpRequest specification the handling betweeen different web programming languages and different browser on different operating systems has become an unified standard. The positive side effects of a web-based discrete simulator using object-oriented simulation paradigm are:

integrity: The script executed on the client side can confirm the existence of a remote object on the web server - and if not so it can initialise it with the provided parameters in the background. Thus the user who is actually dealing with the simulation environment doesn't need to take manual care of the remote object.

usability: An AJAX build web application provides a lot of comfort with higher level user interface designs as for state of the art desktop user interface elements of different graphical widgets can now be integrated in a single web application without rendering heavy loaded images or even embedded objects used by browser plugins.

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 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.

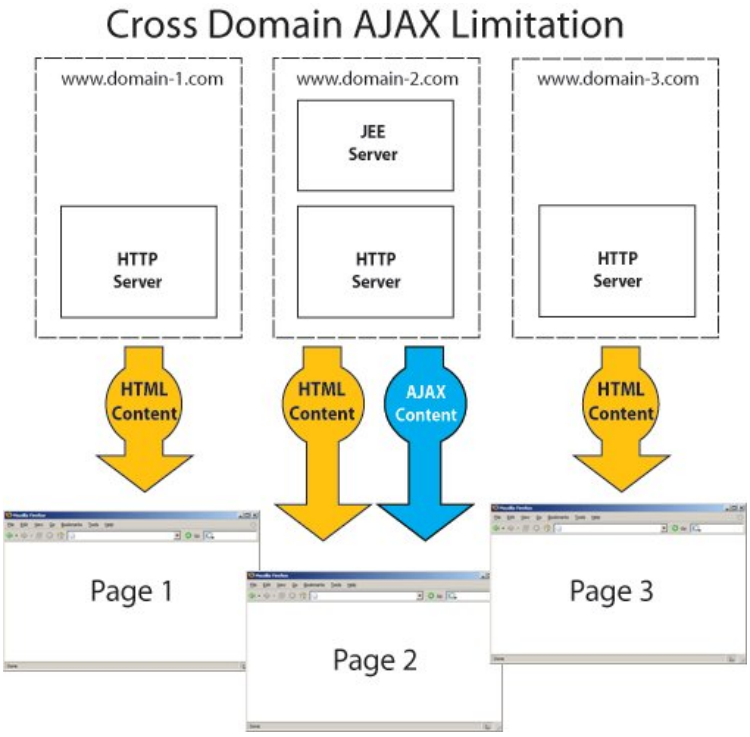

Figure 2: Single domain limitation of AJAX

2.2 Dynamic Code Generation of AJAX scripts

Due to lack of incompatibilities with different web browsers an their interpretation of client sided Javascript code the web application might not work according to the expectation of the application in the server session. Exception handling at many functions and individual browser handling make the source code unreadable and difficult to debug.

A project inaugurated by the open source community and supported by the Google Summer of Code platform named Google Web Toolkit (GWT)[8] has introduced a new approach of Javascript code generation. As for debugging and testing Javascript code is attached with lots of inconsistences, GWT uses a framework where the code is generated dynamically as an output of Java compilation. Thus the development of the simulation web application doesn't aquire manual programming of client sided Javascript code but is is generated with semantic type safety constraints of a compiled language. The development also becomes comfortable for the module developer because all components can be rewiewed and tested within the GWT framework which provides a hosted mode and a browser mode. Furthermore the generated output is eluding browser issues and changes have to be done only in the Java source files. The visual editor in the browser has to provide flexible user interface components for maximum changeability. Hence the purpose of such an interface is a WYSIWYG (What You See Is What You Get) editor for discrete simulation entities. The focus is not mainly on content manipulation and rich text editing (for documentation) but also on a usable graphical interface with a powerful semantics in the background.

3 Working on multiple domains

Working with the web demands transparent resource location with global access or authentification. When the user models the entities, plugs them together and launches triggers for interaction with other entities the resources needed by these entities might not be available on the simulation web server but maybe at some different location. This happens usually when the data is coupled with information retrieved from a database or result files on a file server. Reasons for such scenarios could be:

- Restrictive access to the data
- Processing of the data is done in a different programming language than JAVA
- The objects are too heavy loaded with XML notation which causes latency

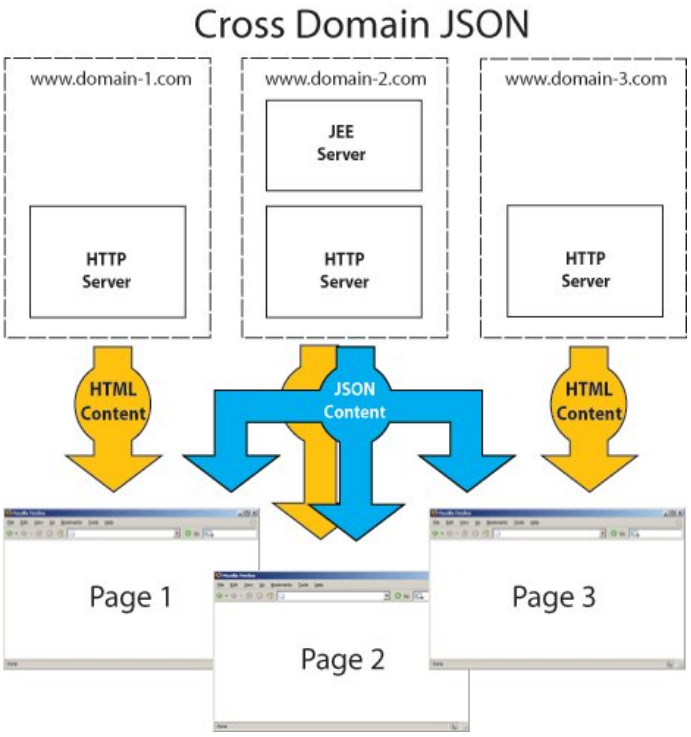

Figure 3: Multiple domain data sharing with JSON

3.1 Sharing objects with JSON

Describing the state of an entity with XML is very common and easy to extract. But XML creates very verbose output which on the otherhand uses more computation time in a browser which has already limited resources. JSON (JavaScript Object Notation) is a much more light weighted alternative where encapsulation and arrangement of data is done with simple, square, and curly brackets.[12]

Figures 3 and 2 show how a JSON enabled web application can share the objects to different sites. The transfer of JSON annotated objects can be passed over the HTTPXmlRequest object. [11] Bindings and libraries for most common programming languages like C, C++, C#, Java, PHP, languages from the .Net suite, etc. provide easy sharing of information within different frameworks and with reduction of overhead tag noise and a better human readability. A sample code for a JSON model represention of a CPU looks like:

```
{
   "vendor_id": "AuthenticAMD",
   "cpu_family": "15",
   "spec": "64b/3500+"
   "flags": [
      {"hw_a": "fpu", "vme", "de", "pse", "tsc", "msr", "pae", "mce", "cx8", "apic"}
      {"hw_p": "sep", "mtrr", "pge", "mca", "cmov", "pat", "pse36", "clflush", "mmx"},
   ],
    "meta_data": [
      {"clock": "200", "unit": 1, "value": "MHz"},
      {"version:": , "15.15.2"},
      {"phyical_id": "integer", "value": "3"}
   ],
   "capabilities": [
      {"type": "string", "format": "alphanum",
       "value": "pipeline-burst internal varies unified", "%s"", "L3-Cache"},
   ],
}
```

As we can see the notation is using a key-value pair syntax and nesting and concatenation for complex data structures. To parse a value out of the model within our script the only code fragment is sufficient:

```
function cpuModelHandler() {
   if (req.readyState == 4 /*completed with parsing*/) {
       var cpuCapabilities = document.getElementById('capabilities');
       var card = eval('(' + req.responseText + ')');
       fieldNameinHTMLForm.value = card.cpuCapabilities[0].value;
   }
}
```

4 Distributed manipulation of objects

In discrete event oriented simulation some tasks are likely to avoid temporal latencies because of quick read and write operations. Thus the current state of the model is required to be broadcasted to all stakeholders in the session. This means that even the client and web server and the application server (which might be configured as a multi tier architecture) need to share the information which describes the state of the entity. In such case the following requirements have to be met:[1]

- Every client on the network and attached to the corresponding simulation session gets an update of all changed variables.
- The manipulation (write operations) of a parameter of the model has to be transactional. Thus the subscribed clients have to receive a call for updating their data and the the web application on the server has to compare the manipulated values. In terms of AJAX scripts this being done with background hand-shakes with acknowledgement responses.

In network-based simulation environments where a inpredictable number of modules work together it is likely that if one component fails all the stakeholders have to react ot the drop out. Thus the design should consider:

Lankiness. The client should be simple and contain only a few classes to decrease overhead traffic.

Awareness. Discrete events triggered remotely have to be recognized and verified in case of data is lost or falsified during transfer.

Fault tolerance. 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.

The simulation web framework is deployed as a J2EE (Java2 Enterprise Edition) application. Persistent data storage for the e-Learning course data (student information, course information, history of the simulation processes, documentation, etc.) are stored in a MySQL database management system. The message handling with the AJAX calls is forwarded to the servlet generated conducted by the GWT. Thus the e-Learning on the server side contributes following services:

- Persistent database
- Non-persistent application cache
- Asynchronous HTTP handling with server sided Java Beans
- Modelling of browser objects with JSON
- internal load balancing prior the actual simulation
- remote execution of a 3rd party simulator via web services

5 Conclusion

This work introduces the web-technological background for an e-Learning simulation service. With the innovative mechanisms of Web 2.0 technology, design issues are discussed how data of entities in a discrete simulator can be passed on to the web server optimizing the data size and reducing computation latency. Now that most programming languages are familiar with the JavaScript Object Notation (JSON) and the overhead for non-verbose data representation is decreased, the data-sharing on the client side is done with JSON objects. From the developer point of view the code generation is being supported with an abstract layer provided by the GWT. The acutal web application is a Java based J2EE application with a persistence database and networking layer.

6 References

[1] Atri, A. and Breitenecker, F. and Nagele, N.: *Distributed Discrete Simulation on the Web*. In: Proc. 20th European Modeling and Simulation Symposium EMSS2008 , Campora San Giovanni, Amantea (CS), Italy; A. Bruzzone, F. Longo, M. Piera, R. Aguilar, C. Frydman; Univ. Calabria, 2008, 392–397.

[2] Chandrasekaran, S.: *Web Service technologies and their synergy with simulation*. In: WSC '02: Proceedings of the 34th conference on Winter simulation, E. Yücesan and C.-H. Chen and J.L. Snowdon and J.M. Charnes, eds.; WSC; San Diego, California, 2002, 606–615.

[3] Gibson, B.: *Enabling an Accessible Web 2.0*. In: W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A), Banff, Canada, 2007, ACM, New York, 2008, 1-6.

[4] Ferretti, S. and Mirri, S. and Ludovico, A.M. and Roccetti, M. and Salomoni, P. Henriksen, J.O. and Lorenz, P. and Hanisch, A. and Schriber, T.J.: *E-learning 2.0: you are We-LCoME!*. In: W4A '08: Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A), Beijing, China, 2008, ACM, New York, 2008, 116–125.

[5] Fishwick, P.A.: *Using XML for Simulation Modeling*. In: WSC '02: Proceedings of the 34th conference on Winter simulation, E. Yücesan and C.-H. Chen and J.L. Snowdon and J.M. Charnes, eds.; WSC; San Diego, California, 2002, 616–622.

[6] Henriksen, J.O. and Lorenz, P. and Hanisch, A. and Schriber, T.J.: *Web Based Simulation Center: Professional support for simulation projects*. In: WSC '02: Proceedings of the 34th conference on Winter simulation, E. Yücesan and C.-H. Chen and J.L. Snowdon and J.M. Charnes, eds.; WSC; San Diego, California, 2002, 807-815.

[7] Rioux, G.P and Nance, R.E.: *Generalizing: Is it possible to create all-purpose simulations?*. In: WSC '02: Proceedings of the 34th conference on Winter simulation, E. Yücesan and C.-H. Chen and J.L. Snowdon and J.M. Charnes, eds.; WSC; San Diego, California, 2002, 783–790.

[8] Google Web Toolkit: Webtoolkithttp://code.google.com/webtoolkit/

[9] W3C web applications: http://www.w3.org/2008/webapps/

[10] W3C XMLHTTPRequest: http://www.w3.org/TR/XMLHttpRequest/

[11] Javaworld: http://www.javaworld.com

[12] JSON: http://www.json.org