Integration of HLA Simulation Models into a Standardized Web Service World

Andreas Wytzisk, Ingo Simonis
Institute for Geoinformatics, University of Muenster
Robert-Koch-Strasse 26-28
48149 Muenster, Germany
+49-251-83-30057, +49-251-83-39763
[wytzisk, simonis]@ifgi.uni-muenster.de

Dr. Ulrich Raape
Fraunhofer Institute for Factory Operation and Automation IFF
Sandtorstrasse 22
39106 Magdeburg, Germany
+49-391-4090-359, +49-391-4090-93-359
raape@iff.fhg.de

Keywords:
HLA, OpenGIS, OGC Specifications, Interoperability, Integration

ABSTRACT: The High Level Architecture still represents the state of the art for the integration of distributed heterogeneous simulation models. However this statement is qualified: Use is restricted to the tight coupling of simulation components. Standardized integration of HLA-based simulations into a Web-based service infrastructure is not possible. No specifications exist, which allow individual federates or entire federations to be usable or manageable in the World Wide Web. “Web-enabling” is therefore a topical issue of research.

In the spatial domain, mature interoperability standards and specifications have evolved in recent years, mainly driven by the Open GIS Consortium (OGC). This article presents an initial solution that tries to bring two individual standardized domains together by creating a standard compliant bridging framework.

1. Introduction

The High Level Architecture specifies a framework for distributed simulations. Distributed simulation is concerned with executing simulations in loosely coupled systems on geographically distributed computers interconnected by a wide-area network such as the Internet (Fujimoto 1999).

The critical deficit in the Internet is the absence of management functionalities that permit a controlled start and termination of federates and federations by external processes. This flaw is mainly rooted in simulator status and behavior not being specified before joining or after resigning a federation. This gap in the HLA interface specifications prevents Web services from completely controlling simulators. Basis management functionalities are only usable after a simulator exists as a physical process and has successfully joined a federation.

Achieving the World Wide Web’s usability of distributed, HLA-based simulations means being able to control them and make use of the simulations results produced. Therefore this paper will present a concept that allows
- external initiation of federations,
- controlled start up and termination of federates by external management processes,
- interactions with running federates,
- access to simulations results by external, non-HLA-processes.

What makes this approach so interesting is that it is completely based on the IEEE 1516 specification. All extending concepts are grounded in the use of specific FOMs and do not break with the standardization but are all HLA compliant. Necessary additional functionality will be represented by corresponding object and interaction classes within the FOMs and therefore follows the HLA guidelines.

The basic concept behind the external control of HLA federations is the use of so-called management federations, which allow the instantiation and control of so-called simulation federations and their corresponding federates (Housand, 2000 #230 and Lantzsch, 2001 #140).
“Simulation federation” describes a federation that executes the real simulation experiment.

All extensions requiring platform-specific or other specific adaptations are encapsulated in application-independent federates. Once specified or implemented, they can be employed on a universal level (see figure 1).

<table>
<thead>
<tr>
<th>DCP-specific access layer (federation controller service)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management Federation</td>
</tr>
<tr>
<td>Simulation Federation</td>
</tr>
</tbody>
</table>

Figure 1: Layered architecture for external federation control (Wytzisk 2003)

The specifications of the Open GIS Consortium (OGC) suggest themselves as a starting point for integrating simulation services into a standardized Web service world. As an international consortium of more than 230 participating companies, government agencies and universities, OGC is the global leader in developing publicly available interface specifications to “geo-enable” the Web. In one of OGC’s newest initiatives, (Open Web Service Initiative 1.2), Sensor Web Enablement specifications (SWE) have been developed that allow the standardized integration of sensors into the Web service domain. An obvious starting point for integrating dynamic capabilities (simulation) into the Web service world was the awareness that, in terms of the provision of spatio-temporal data, a simulator does not differ from a sensor (it only differs in the way it estimates the requested value and its virtually temporal independence). An in-depth description of this approach can be found in (Simonis et al. 2003).

2. Management and Simulation Federation

A management federation consists of a quantity of federates belonging to one of three different types (see figure 2). Management federates are assigned to one or any arbitrary number of computers. After receiving the corresponding interaction, these federates are enabled to start simulation federates on the local or on a remote machine. To be able to reuse existing HLA-compliant simulators, it is important not to make any demands on their implementation, meaning they have to be executable as self-sufficient programs within their individual process space. To make use of the publish-subscribe paradigm, simulation federates that are executable by management federates are described using federate descriptors. These descriptors are published in the management federation as object instances. The corresponding object class definitions are specified in a Management FOM that is independent of the application domain.

2.1 Federate Types

Management control federates are enabled to publish control commands as interactions. Management federates receive and interpret these interactions. The management FOM defines the syntax and semantics of control commands as well.

Management control federates provide interactions to realize functionalities to start and kill individual federates or federations, and to interact with running simulation federates. A mechanism must be provided that allows management federates to decide which federates could be started and might be merged into a common federation of a particular type. This is done by subscribing to federate descriptor objects.

The documentation procedure defined in the HLA specifications (FOM, SOM) lacks any semantic formulation interactions or object classes. Therefore management federates do not have any possibility to decide which combination of simulation federates will form a practical and profound federation. To deal with

1 Not to be mistaken as the Management Object Model (MOM) specified in IEEE 1516.
2 The semantic meanings of FOM elements are formulated as simple text and therefore cannot be evaluated automatically.
this shortcoming, management control federates are associated with a specific federation type, i.e. a quantity of simulators known a priori. Hence a semantically incorrect federation can be avoided in the course of the implementation or configuration.

Management federates and simulation federations commonly exist in distinct process spaces. Except for proprietary channels of communication, there is no way to arrange any kind of interaction. In particular, this precludes a command to execute a supervised simulation run termination. Paired with the problematic nature of the missing standardized channel of communication, the demand to control a simulation federation necessitates a further federate type, the specific management interaction federate. This federate is tightly coupled with another specialized federate type, the management bridge federate (see figure 3).

All control commands or call back interactions published or subscribed by a management interaction federate are integrated in a simulation interaction FOM. This FOM must be supported by any simulation federate that takes part in the system described.

By virtue of its tight coupling with a management bridge federate, the management interaction federate allows:

- making information about simulation federation participants available within the management federation and consequently to external processes,
- sending user-specific interactions into the simulation federation in order to allow interactions between external processes and running simulations by management control and bridge federates,
- establishing federation execution in particular and therefore providing the only mechanism for reporting a successful or failed start back to the management federation.

Figure 4 illustrates the typical interaction sequence for starting and terminating a simulation federation. It is based on the specific federates described above.

Except for the communication between management bridge federate and management interaction federate, all interactions presented are based on the publishing and processing of HLA interactions. For the sake of clarity this fact is not shown explicitly in the UML diagram.

Beginning with a user request (1) a management control federate sends an interaction to create a management interaction federate (1.1). This is processed by a management bridge federate, which subsequently instantiates a management interaction federate (1.1.1). Afterward, the management interaction federate created starts the federation execution process (1.1.1.1), where the simulation federation is coordinated. After the management interaction federate becomes a member of the federation execution process (1.1.1.2), a callback to the management bridge federate ends up informing the management control federate of the success of this procedure.

When both the management interaction federate and the federation execution process are created successfully, it is time to start the actual simulation federates. Therefore a
specific start interaction is sent (1.2) by the appropriate management federate (1.2.1). This start interaction is loaded with a well defined set of parameters and has the unambiguous name of the simulation federate to be started.

Along with the management federates, the bridge federate also receives the interaction intended to arrange the start of the simulation federate (1.3). Hence the bridge federate instructs the management interaction federate (1.3.1) to examine the federation execution process until the simulation federate has joined the simulation federation (1.3.1.1). In the case of a successful or failed join, a corresponding callback interaction will be sent to the caller.

After the simulation federate has started and has joined the federation execution, it is implementation-dependent if it starts immediately, after synchronization with further simulation federates or after being triggered externally with its actual simulation run. Section 3 explains how the simulation results are made accessible to external processes.
If the simulation federation is terminated early or no abort criteria are defined that dispose federates of the simulation federation to end their simulation runs and finally terminate the federation execution process, the management control federate can be used to enforce the termination.

The management control federate forwards this request to the management bridge federate which by itself disposes the management interaction federate to send the specific simulation federate termination interaction as specified in the simulation interaction FOM.

Each federate of the simulation federation receives this interaction and reacts by resigning from the federation and then terminating its processes. The management interaction federate takes note of the federation resignations and itself resigns the moment it is the last remaining federate in the simulation federation. Then it terminates the federation execution process and finally itself.

### 2.2 Federation Controller Service

In order to allow external processes access to the simulation federation control functionalities provided by the management federation, it is necessary to specify an access layer. This access layer is realized by a DCP-specific, i.e. Web-based federation controller service (figure 5).

*Figure 5: Federation controller service dependencies in UML notation [Wytzisk, 2003]*

Federation controller services aggregate one or many management control federates and provide the following elementary operations:

- **getCapabilities** return an XML encoded document to the caller, which contains information about the federation types supported. In particular, what operations callers have at their disposal and how these operations have to be parameterized are documented. Additionally, an XML schema of the parameterization of the simulation federates is referenced.

- **startSimulation** is used to start a simulation federation. Apart from the definition of the federation type to be started, the request has to be completed with the parameterization of the simulation federate. The return type provided for the caller contains an unambiguous ID that allows further interaction with the running federation. In the case that the federation controller service is aggregating more than one management control federate controlling the same federation type, efficient load balancing depends on the individual implementation.

- **stopSimulation** terminates a simulation federation explicitly, i.e. it initiates the controlled termination of all joined federates and the federation execution process itself. The requestor receives acknowledgement of a successful or failed operation.

- **sendInteraction** (optional) is used to send interactions into simulation federations. A management bridge federate facilitates the dispatch of the interaction the requestor must parameterize. The federation controller service is not part of the simulation federation and therefore not integrated in the time advance procedure of the simulation federation. This restricts the interactions to “receive-ordered” (RO) only. These interactions are explicitly declared in the simulation federation FOM and documented in the “capabilities” document of the federation controller service.

- **getObjectInstances** (optional) returns an XML encoded document that describes all instances of a specific interaction class observed by a management bridge federate. The current status of a running simulation can be requested this way. All observable object classes are defined in the simulation federation FOM and documented in the capabilities document of the federation controller service.

- **getObjectInstance** (optional) reflects the current status of an object instance at the time the management bridge federate receives the request. The requestor has to know the unambiguous ID in order to be able to access a single object instance. This ID could be obtained by executing a “getObjectInstance” operation.

---

3 Instructing the last resigning federate to terminate the federation execution process common practice as part of the implementation of HLA federates.

4 As opposed to the stopSimulation operation, it is possible to realize federations that terminate when an abortion criteria becomes true.
modifyObjectInstance (optional) allows the modification of an object instance, meaning changing the attribute values. As is the case when dispatching an interaction, this operation only permits a “receive-only” mode.

3. Provision of simulation results

Basically, two different approaches to providing simulation results must be distinguished, primarily by invoking the federation controller service interface operations “getObjectInstance/getObjectInstances” already described. These operations allow the accessing of instances of published object classes in a simulation federation. These methods have the disadvantage of being restricted a single moment in time: Accessing the actual object states of the running simulation is possible but accessing any kind of historical data is not.

To enable access to actual as well as to historical simulation results, a further specific federate must be integrated into the simulation federation: the simulation listener federate. This federate makes all instances and instance modifications, which it has subscribed according to its SOM, available in a persistent database.

A public Web interface can access the content of the database (Figure 6). To produce a generic simulation listener federate, it is necessary to create a data model that defines the structure of the subscribed object classes as well as the corresponding instances. This generic simulation listener federate can be used independent of the application domain.

Figure 7 illustrates such a data model. Entity types (gray background) describe the object classes (ObjectClass, ObjectClassAttribute) subscribed by the simulation listener federate as well as the data types used (BasicDatatype, Datatype). The latter might be simple data types (double, integer) or complex. Complex data types are modeled using a self-referencing entity type Datatype in a tree structure.

The ER diagram uses entity types with a white background (ObjectInstance, ObjectInstanceAttribute) to depict published object instances in a specific simulation federation. The status of published object instances is commonly updated during a simulation run. Therefore the administration of the real attribute values (ObjectInstanceAttributeValue) is modeled as a function of the federation-specific time stamp.

Attribute values might have a complex data type. If this is modeled in a tree structure (as it is here), only the last leaves of the tree will hold real data values. All other leaves are limited to structural functions.

To persistently save arbitrary simulation results independent of the application domain, the simulation listener federate must be able to:

- read the accompanying SOM automatically,
- enter the SOM defined structural information about subscribed object classes into a database,
- track and save publications and modifications of object instances after joining a simulation federation.

5 This is the moment in time when the management interaction federate is requested to publish the reflection of an object instance into the management federation. Due to the latency time indeterminable a priori between the federation controller service request and the subsequent reaction of the management interaction federation, it is impossible to determine the time stamp of the returned object instance (independent of the relation to simulation time or real time).
The common data model enables application developers to create access methods with a high level of reusability. These methods can be invoked behind standardized service interfaces.

Apart from the possibility described of making simulation results available after a user invoked request, it is also possible to actively inform a user about new simulated data. The basis is an extension of the simulation and management federations (including simulation listener federate and management interaction federate), which initiates a message when previously defined object instances are published or modified. The message might contain a reference to the result or the result itself. If the receiver is a service, it is possible to use service chains triggered by simulation results.

4. Conclusions

The “management federation” concept is a promising approach to make HLA federations controllable from the World Wide Web without needing to extend current standards. A first prototypical implementation has verified this. Future research work will have to focus on the specification of a simulator and the parameterization of description language. This would make the “service trading” concept for Web services possible even for simulations. First steps are being taken by (Kilgore 2001; Wiedemann 2002) and will be studied in the recently established SISO working group “simulation reference markup language” (Reichenthal 2002a) (Reichenthal 2002b). SWE provides a good starting point for developing a closer approach to both the simulation and geo-spatial domains.

References


Author Biographies

ANDREAS WYTZISK is a project manager at the Institute of Geoinformatics at the University of Münster, Germany. He studied Physics and Geography in Heidelberg and Münster. In 1996 he co-founded the company pro_Plant for agricultural and environmental informatics. He completed his doctoral dissertation in early 2003. His research topics include the integration of service-based geographic information infrastructures and simulation-based software architectures as well as logistical problems in agriculture and forestry.

INGO SIMONIS is a project manager at the Institute of Geoinformatics at the University of Münster, Germany. He holds a master’s degree in Ecology and is currently working on his dissertation, issuing the usability of simulation results in a Web service environment and subsequent integration into a Web-based decision support framework. His further research focuses on the integration of service-based geographic information infrastructures and simulation-based software architectures as well as mobile computing concepts, architectures, and techniques.

ULRICH RAAP is a project manager in the Department of Environmental Engineering at the Fraunhofer Institute IFF. He received his Ph.D. from the Otto von Guericke University in Magdeburg, Germany. He holds a master’s degree in Industrial Engineering from the University of Karlsruhe and worked two years as a project manager for Command, Control and Communication Systems for Public Safety and Security in Europe. His research topics include Emergency Management, logistics, Geographic Information Systems and HLA-based software architectures.