Proceedings of the 6th AGILE
April 24th-26th, 2003 – Lyon, France

INTEGRATING SIMULATION MODELS INTO SDI'S

Ingo Simonis, Andreas Wytzisk, Ulrich Streit

Institute for Geoinformatics, University of Münster
Robert-Koch-Str. 26-28, D-48149 Münster, Germany
e-mail: {simonis, streit, wytzisk}@ifgi.uni-muenster.de

Abstract

Spatial data infrastructures (SDI) are currently one of the hot topics within the geographic information domain. Mature interoperability standards and specifications have evolved in the last years, mainly driven by the work of the OpenGIS Consortium (OGC)[1]. As there are a couple of specifications available describing the interchange and processing of static geographic data and forming the basis for building up a SDI, the demand of accessing spatio-temporal data via the web is growing but is still unsatisfied. First actions have been taken by standardizing interfaces for the "Sensor Web Enablement" (SWE), that allows near realtime access to georeferenced measurement data collected by sensors. Beeing aware of the fact that a simulator does not differ from a sensor regarding the provision of spatio-temporal data (it only differs in the way how it estimates the requested value and its virtually temporal independence), there is an urge to include simulators into SDI's which would fulfil the need of countless application domains: It would provide sophisticated simulation functionality via an easy to use web-based infrastructure. Rather than inventing the wheel again, building bridges between OGC specifications and standards within the modelling and simulation community will allow both communities - the geoinformation one as well as the simulation one – to benefit from what has been achieved. This paper describes the current results of a project aiming at the interoperability of the leading standardized simulation framework, the High Level Architecture (HLA) and the OGC specifications. The functionality of simulations will be made available to the web service community.

1. INTRODUCTION

Geographic information has per se spatial, temporal, and thematic aspects (Ott und Swiaczny 2001). Therefore, future SDI-functionalities implies not only the accessibility to spatial information, but also assumes a mandatory access to spatio-temporal information about real and virtual processes. There exist countless examples of possible SDIs showing the necessity to overcome the status quo which allows only the use of rather static information, e.g. emergency management, telematics and logistics, ecological monitoring. It is demanded to integrate monitoring and simulation capacities. The reasons are almost obvious: Imagine the first mentioned application domain, the emergency management. Almost three billion Euro of damage had been caused by the last floods of the Elbe River in the eastern parts of Germany just one year ago. It was nearly impossible to get any valuable calculation about the development of the water masses. It would had been very helpful if the emergency management operators had been known where the people had to be evacuated at what time, where the dams are not powerful enough to hold the water or what would be the effect if a dam would be opened at a certain location to protect another more valuable area from flooding. At least partly these questions would have been answerable using simulation functionalities, coupled with static data that is already stored in data bases. It is absolutely necessary to provide simulation capabilities to the web: Simulations that are continuously running and provide a calculation for a moment in the future, and those

simulations which have to be started individually after feeding with the mandatory parameters. These simulations will allow a scenario calculation and answer the question: What would be if…? The OGC *Sensor Web Enablement* provides the architecture and infrastructure to solve these demands.

2. SPATIAL DOMAIN

Current SDIs are mainly based upon specifications developed by the OpenGIS Consortium (OGC) which was founded as a non profit organisation in 1994 and has currently over two hundred fifty members. Driven by the most important GIS, Operating System (OS) and Database Management System (DBMS) developers, vendors, and users, it is now one of the world leading organisations addressing interoperability issues within the spatial domain. The interoperability platform for the geo-processing domain used in this paper is based on the OpenGIS specifications provided by the OGC.

Until last year, developments within OGC mainly focus on the interoperability of data accessing and processing capabilities and the representation and interchange of geographic data itself. Both was made available to the World Wide Web by standardizing web interfaces. These unique service specifications will lead to a global geoinformation infrastructure. Interoperability at the data level is achieved by defining the Geography Markup Language (GML) as an XML encoding for the exchange of geographic information, including both the spatial and non-spatial properties of geographic features [2].

As there are a couple of specifications available describing the interchange and use of static geographic data, the demand to access spatio-temporal data and operations via the web is growing. Within the spatial domain, first actions have been taken by standardizing interfaces for the “Sensor Web Enablement”.

2.1 OGC - Sensor Web Enablement

The “Sensor Web Enablement“constitutes the OGC approach to make sensor data available via web services. It was initially designed to fulfil the following needs:

- Describe sensors in a standardized way;
- Standardize the access to observed data;
- Standardize the process of what is commonly known as sensor planning, but in fact is consisting of the different stages planning, scheduling, tasking, collection, and processing;
- Building a framework and encoding for measurements and observations.

The demands expressed in these points had lead to the development of four different specifications which will be described in detail. To describe sensors, the markup language *SensorML* had been created. The service interface specification of a *Sensor Collection Service* provides access to observed data whereas the *Sensor Planning Service* provides an interface to organize the observation of natural phenomena. The results of these observations are XML encoded following the standard described in

Observation & Measurement.

During the development of the specifications mentioned above, it became obvious that sticking to the synchronous character of OGC web services was not possible any more. The execution of complex observations (like ordering satellite images from a certain location with maximal cloud coverage of five per cent) or complex simulation runs may take virtually any

time to fulfil. For this reason, the authors created a new service specification that allows asynchronous service handling: the statefull *Web Notification Service*.

Sensor Collection Service

The objective of OGC's SWE activities is to develop a set of interfaces to access a single or a constellation of real-time sensors heretofore referred to as a *Sensor Collection Service* (SCS) [3]. Once accessed, a SWE client application receives readings from a sensor via the SCS where it displays or processes the data in a variety of manners.

Sensor Planning Service

The Sensor Planning Service (SPS) is intended to provide a standard interface to collection assets (i.e., sensors, and other information gathering assets) and to support systems that surrounds them [4]. A SPS not only has to support different kinds of assets with differing capabilities, but also different kinds of request processing systems, which may or may not provide access to the different stages of planning, scheduling, tasking, collection, processing, archiving, and distribution of resulting observation data. The SPS is designed to be flexible enough to handle such a wide variety of configurations.

Web Notification Service

The WNS is an asynchronous and statefull service. It is a web interface (e.g. operated by the SPS) that allows sending notifications to a client with well structured content. To enable any kind of dialogue between the user and an invoking service, functionality has to be provided that enables the user to answer asynchronously with any kind of structured content. Currently four different notification delivery channels are supported: e-mail, SMS, HTTP POST, and instant messages. It had been avoided to specify a "SWE-WNS" rather than a WNS, because web notification functionality will become necessary for other web services as well at the moment SDI usage will become real business [5].

SensorML

SensorML is a conceptual model that describes the geometric, dynamic, and observational characteristics of a sensor using XML encoding. Sensors are devices for the measurement of physical quantities. There are a great variety of sensor types from simple visual thermometers to complex earth observing satellites [6].

Observation and Measurement

The Observations and Measurements Engineering Specification describes a conceptual model that allows the depiction of spatial and temporal variant information, which is the result of the estimation of some natural phenomenon. The representation is based on the specified XML encoding. For Cox, measurement usually refers to the measuring device and procedure used to determine the value, such as a sensor or observer, analytical procedure, simulation or other numerical process. The carrying out of the procedure to estimate the value of the phenomenon is called observation [7].

Figure 1 illustrates an UML representation of an observation feature, which defines an observation that is bound to a location, a time, and a procedure:

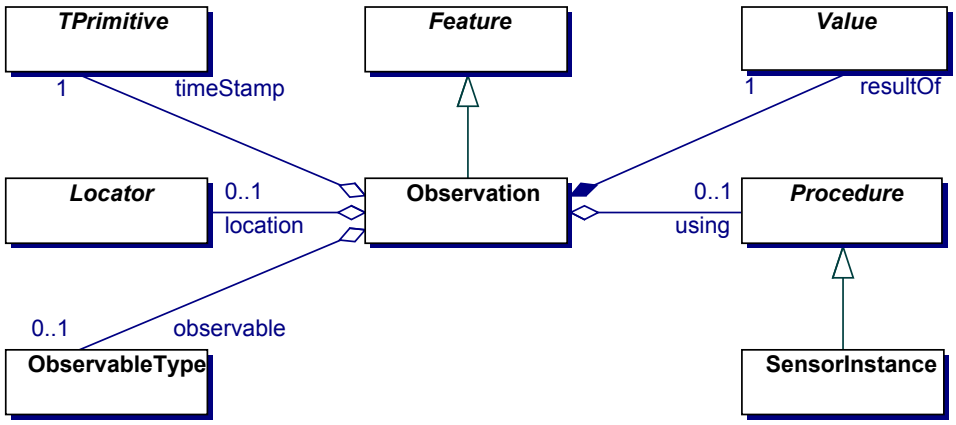

Fig. 1 UML representation of an observation feature

2.2 From Sensor to Simulator – the Extended SWE

The efforts taken so far had been extended by the authors to allow the use of simulators instead of sensors. Being aware of the fact that a simulator does not differ from a sensor regarding the provision of spatio-temporal data (it only differs in the way how it estimates the requested value and its virtually temporal independence), two different scenarios can be distinguished. In the first case, the simulator is continuously running, independently of the user. It is started, parameterized and maintained by the data provider and has no interface to the user other than the pure data access. These simulators are e.g. weather forecasts that provide values for the parameters temperature, precipitation and speed of wind or air or water quality assessments. It is even possible that a user will not notice at all that the provided data is resulted from calculations rather than real measurements (e.g. if the values are interpolated due to an rather wide-ranging measuring network). Those simulators will be encapsulated to a simple datastore that is accessible using the Sensor Collection Service. It can be handled purely synchronous and follows the service trading (publish – subscribe – bind) paradigm which supports the dynamic binding between service providers and requestors:

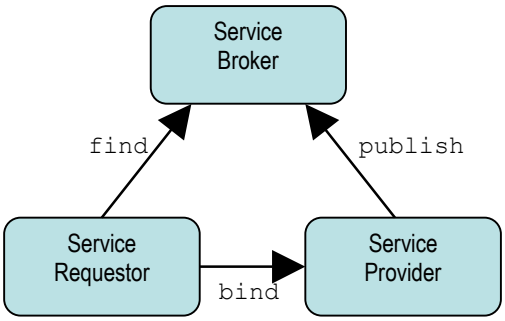

Fig. 2 Service trading by a publish-find-bind mechanism

The service provider publishes services to a broker (registry) and delivers binding information to service requestors. Let's imagine an air quality sensor provider. It publishes its Sensor Collection Service to the service broker. A service requestor, in our case a user

who is interested in air quality measurements performs a service discovery operation on the service broker to find the service providers which provide the measurements he needs. The service broker acts as a registry or clearinghouse of services and helps requestors and providers to find each other. The user finds the Sensor Collection Service and accesses the data. The scenario becomes more complex, if – for example – the user is a car traffic operator and just needs to get informed in case air quality goes beyond a critical level.

All observations or simulations that require preceding feasibility studies, complex control and management activities, or intermediate and/or subsequent user notifications, are not manageable synchronously anymore but become heavily asynchronous. In this second case the service interactions become much more complex. Let's continue the previous example and extend it with an accident taken place in a local chemical plant. The traffic operator will compile the best routes for car drivers by means of avoiding contaminated areas as best as possible. The contamination plume is simulated by a dispersion model. He uses two web services, a Sensor Planning Service as an interface to the otherwise black boxed simulation engine, and a Sensor Collection Service that allows the further data access. To notify the user when the requested data is available, he uses a Web Notification Service from another provider. The scenario will follow the sequence shown in figure 3:

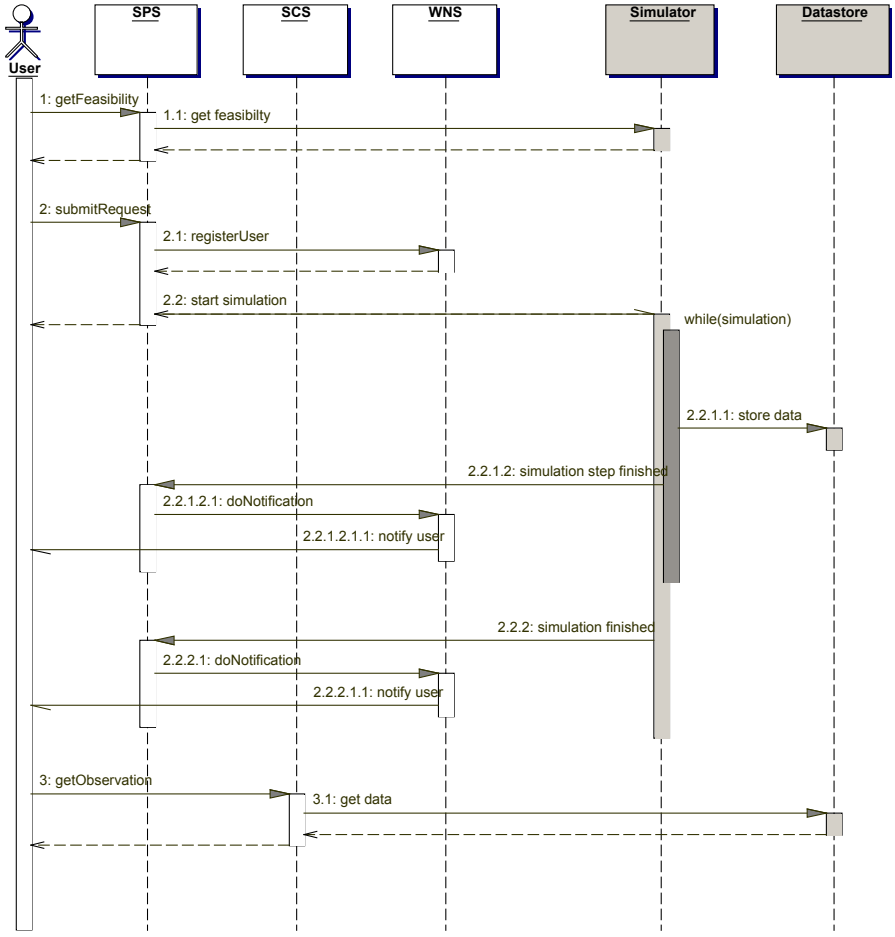

Fig. 3 SWE usage sequence

First the user investigates if the required data might be available for his location by sending a getFeasibility request to the Sensor Planning Service (to simplify the sequence, it is assumed that the user knows the location of the service). Depending on the capabilities of the simulator (supposing that the simulator has static base data and dynamic wind field data available), the user receives a feasibility response accompanied by a XML encoded form. The form contains fields for the necessary parameters the simulator needs to start the simulation run, e.g. the location of the accident, the channel used for notification (email, SMS, etc.) and so for. Afterwards he submits the request. Due to the fact that the calculation takes some minutes to be completed, a Web Notification Service is used to notify the user when the results will be achievable. The user receives a notification that the simulation run has started together with a UserID, a WNS URI, and a TaskID that allows process mapping. Depending on the user defined parameters, he will receive notifications whenever a simulation step is completed or only once when the simulation run is finished. The data is stored into a data store. The Sensor Planning Service notifies the user about this event and provides an URI of a Sensor Collection Service that allows access to the data.

In this scenario most of the processing and data retrieving is black boxed. Certainly it would be possible to provide an SCS URI where the dispersion model can get information about the current wind field (which again could be provided by another simulator).

3. SIMULATION DOMAIN

Regarding the integration of distributed heterogeneous simulation models the specifications of the High Level Architecture (HLA) are still state of the art [8]. The HLA framework, born in the military domain in 1996, has broken into the civil domain during the last five years and constitutes the only standardized simulation framework so far (as an advancement of DIS [9]). A distributed simulation that is based on the HLA specifications is called a federation, where the distinct members are encapsulated as federates (members could be any kind of simulator, live player, etc.). All federates of a federation are communicating via the RTI (runtime infrastructure), a kind of a qualified data bus. There is no direct inter-federate communication allowed. The federation possesses of a *federation object model* (FOM) which contains the description of the object and interaction classes that are utilizable for all federates within the federation.

To attain availability of simulation models using a Sensor Planning Service, it becomes necessary to specify a HLA based framework that allows the management of any distributed simulation run from any process, independently of the kind of federates of federations. This management functionality is not part of the standard IEEE 1516: Status and behaviour of a federate are not defined before it joined respectively after it left a federation. There are basic management capacities available only in case a simulator already exists as a physical process and has already joined a federation. But even these management capacities are not usable from external processes like web services but are only to the federate's disposal.

To make HLA based simulation available via WWW, means to control the simulation run and to access the results, we will show a concept that will allow the following points without infringing the IEEE 1516 specification:

- To initiate federations externally
- To control start and termination of federates using an external management process
- To make the simulation results available for external, non HLA-processes

On the one hand, the enhancements are based on the concept of a *management federation* that allows instantiation and management of the actual *simulation federation* and

corresponding federates. “Simulation Federation” defines the federation which actually executes the simulation experiment. On the other hand, they are based on a specified reference FOM.

A management federation consists of a bulks of federates. This federates belong to one of the four types:

- Management federates
- Management control federates
- Management bridge federates
- Management interaction federates

Management Federates are responsible to start simulation federates locally or remotely. The simulation federates are described by federate descriptors that are published within the management federation. The object class definitions are part of the *Management FOM*, which itself is independent of the simulation domain.

Management Control Federates send control commands as interactions into the federation – e.g. to start or terminate simulation federates and simulation federations – which are received and processed by the management federates. Syntax and semantic of the control commands are defined in the FOM as well.

Usually, management federates and simulation federates exist as distinct processes and are not able to communicate except of using proprietary channels. To be able to control a simulation federation, we will have to extend it by a specific *management interaction federate* that is closely coupled to a specialised management federate, the so called *management bridge federate*. This management interaction federate sends commands to the simulation federates, e.g. for termination purposes. All control and callback interactions that are subscribed by the management interaction federate are described in a simulation interaction FOM that has to be supported by all simulation federates that will participate at the described practice. Figure 4 shows the different federate types in UML notation.

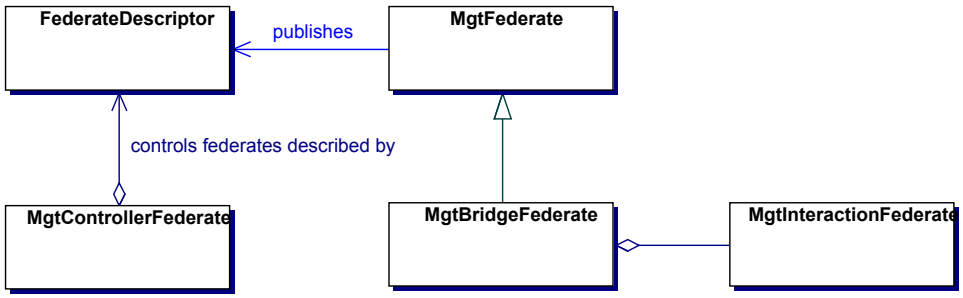

Fig. 4 Federate types in UML notation

As a result of the close coupling with the management bridge federate management interaction federates allow to:

- Retrieve information about the members of a simulation federation from the management federation and therefore make this information available to external processes.
- Send user-specific interactions into the simulation federation. Hence it becomes possible to control a simulation federation from external processes by make use of the management control federate – management bridge federate bridge.

4. CONCLUSION

The integration of simulation models into spatial data infrastructures was shown. The concept is based two state-of-the-art interoperability technologies (OGC for the geoinformation domain and HLA for the simulation domain).

By specifying and extending the Sensor Web Enablement ambitions by integrating simulation models, a complete new improved category of SDIs was enabled. Near real time data is made available to end-users as well as the access to different scenarios, individually calculated for each user and accessed via standardized interfaces.

This paper presents ongoing research. First prototypes and usage concepts are shown at the OGC Open Web Service Initiative Demonstration Meeting in Washington in November 2002 with great success. All further developments will be integrated in the ongoing specification work of the OGC.

5. REFERENCES

[1] OGC(2002): Open GIS Consortium, http://www.opengis.org

[2] Cox, Simon, et al.(2001): Geographic Markup Language GML. OGC Recommendation Paper.

[3] McCarty, Tom(2001): Sensor Collection Service. OGC Interoperability Program Report-Engineering Specification.

[4] Lansing, Jeff(2002): Sensor Planning Service. OGC Draft Interoperability Program Report-Engineering Specification.

[5] Simonis, Ingo; Wytzisk, Andreas(2002): Web Notification Service. OGC Draft Interoperability Program Report-Engineering Specification.

[6] Botts, Mike(2002): Sensor Modelling Language. OGC Interoperability Program Report-Engineering Specification.

[7] Cox, Simon(2002): Observation and Measurement. OGC Interoperability Program Report-Engineering Specification.

[8] IEEE(2001): High Level Architecture. IEEE 1516.1/2. IEEE Specification.

[9] DMSO(1998): High Level Architecture. Master Plan. http://www.dmso.mil.