Jul 1st, 12:00 AM

Model Coupling and Integration via XML in theM3 Simulation

Andreas Hoheisel

Model Coupling and Integration via XML in the M3 Simulation

A. Hoheisel

[a] *Fraunhofer Institute for Computer Architecture and Software Technology*
Kekuléstraße 7, D-12489 Berlin-Adlershof, Germany
andreas.hoheisel@ rst.fraunhofer.de

Abstract: Complex real-world systems are currently developing into a decisive instrument for IT-supported problem solving for a great number of problems posed by science, economy and society. Like in many environmental simulation systems the Man Model Measurement (M3) Simulation (http://mmm.first.fraunhofer.de) contains numerous single scientifically founded models coupled to each other to reflect the complexity of our world. The unpredictable human factor is considered by actively involving real people into the simulation by a Virtual Reality framework. The evolution and enhancement of the M3 system will be provided by an open software API. This paper will focus on the control and the data communication between the distributed simulation models that are realized by specialized markup languages which employ the XML technology for easy extension and validation. The input and output data of the models is represented by an XML data model that includes also the physical and mathematical meaning of the transferred data so that different models can be easily coupled by connecting their XML formatted input and output streams, e.g. via sockets. The initialization parameters and the relevant output data of the models can be stored in an SQL database (MySQL) which can be accessed by a web interface or directly using JDBC. Legacy Code is included in the simulation system using wrapping technologies.

Keywords: Model Coupling; Model Integration; XML; Actor-based Simulation; Virtual Reality

tion systems with the multimedia possibilities of virtual reality. Hereby, the idea of embedding scientifically founded simulation models into a Virtual Reality Environment (VR) that involves real human actors in the simulation is essential. Thus we get a scientific instrument for the assessment of complex systems involving human behavior that can be helpful for the development of sustainability strategies and serve as a decision support for politicians. In addition, this system has also a didactic function because the people acting in the M3 simulation, the so-called actors, are confronted with the consequences of their actions in quick-motion. Further considerations about the principle ideas of the M3 simulation can be found in the paper of Rosé [2002] in this volume.

The general structure of the M3 system is unified by a Multi User Virtual Environment (MUVE) representing the logical structure of the simulated world and coordinating the communication between the three main components of the framework (Jugel [2001]): the Multi-Purpose Graphical User Interface for visualization of the simulated world in specialized views adapted for the different user groups (actors, experts, decision makers, publicity), the Model Server providing a distributed network of scientific simulation models connected via generic control and data mapping interfaces (RMI, XML), and the Measurement Database providing current measuring data, reference configurations and parameters of the real environment in question. The most important user group for the success of the M3 simulation is the group of actors who are represented by so-called avatars (Figure 1). Figure 2 shows a snapshot of the M3 expert client that is used for controlling the simulation.

A major task of the M3 simulation is the integration of several scientific models developed by external institutes or by ourselves. Each model simulates the processes of a certain part of the environment and is maintained by the specialists who were also responsible for the development of the model. For this purpose we build a component framework consisting of interface definitions and protocols for the communication between the components of the M3 system.

2 Models used in the M3 Simulation

Virtual worlds have no relation to reality without the implementation of scientifically validated models. Because of the complexity of the system it is not practicable to simulate all classes of environ-

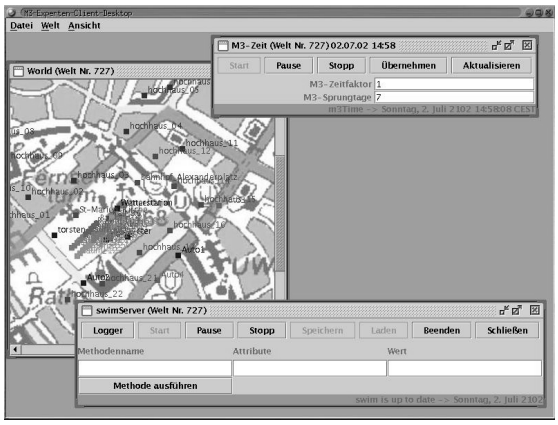

Figure 2. M3 expert client with control panels for each simulation component

mental processes with one monolithic program. In the M3 system a model server is used to provide a distributed network of scientific simulation models connected via generic control and data mapping interfaces.

Currently we include the following models in the M3 simulation framework:

- SWIM (Soil and Water Integrated Model) — integrates hydrology, vegetation (e.g. crop growth), erosion, and nutrient dynamics at the watershed scale. SWIM was developed by the Potsdam Institute for Climate Impact Research (Krysanova et al. [1998]) (programming language: Fortran77, Fortran90)
- 4C — a model to simulate the growth of trees in the forest including hydrology, developed by the Potsdam Institute for Climate Impact Research (Bugmann et al. [1997]) (Fortran90)
- Wgen — a weather generator developed by Richardson and Wright [1984]. On the basis of statistical data, it produces daily weather by a random process (C)
- REWIMET — a hydrostatic, three-layer model for the propagation of gases in the atmosphere including the current meteorological situation. The model is coupled with a Lagrangian model to simulate the propagation of PM_{10} particles (Unger et al. [1998], Mieth et al. [1998]) (C)
- Impact — a model to simulate the impact of the environment on the health of the people

due to pollution of air, ground and food developed by Beger et al. [2001] (Java)

- Mobility — a mesoscopic traffic simulation model developed by Fraunhofer FIRST (Schmidt et al. [1998]) (Java)

A general problem that is addressed here, is the integration of legacy code. Most of the models in the environmental domain are normally used as standalone programs on a specific platform. The models are generally not designed to be used in a coupled simulation in conjunction with other models that may use another programming language or even require a special operating system. The M3 framework is based on Java, and we use Java wrappers for the simulation programs that are written in other programming languages, in order to mask the differences in the implementation.

3 Model Coupling

The M3 simulation is an open framework in the sense that other models can easily be connected to the system if they implement the protocols and interfaces supported by the M3 component framework or if there is a suitable Java wrapper that maps the data structures and method calls used in the M3 system to the proprietary syntax of the simulation model.

In most cases a tight coupling scheme is used in environmental simulation, and shared memory is used for the communication between the coupled models. Tight coupling generally requires a lot of effort when integrating the models, and often it is necessary to implement all models in the same programming language. In the M3 simulation we use a loose coupling scheme instead, with asynchronous communication between the coupled models. The model coupling is network-oriented and based on semistructured or unstructured data exchange formats (e.g. XML). This kind of model coupling is very flexible and makes it easy to reuse single models in another context.

In the M3 simulation we use dynamic coupling with late time binding during runtime, so that the coupling does not have to be defined when compiling. This enables us to select or exchange simulation models during runtime without the necessity of restarting the simulation or recompiling the simulation program. The models of the whole simulation are distributed over several concurrent processes that use online communication for bidirectional or unidirectional coupling.

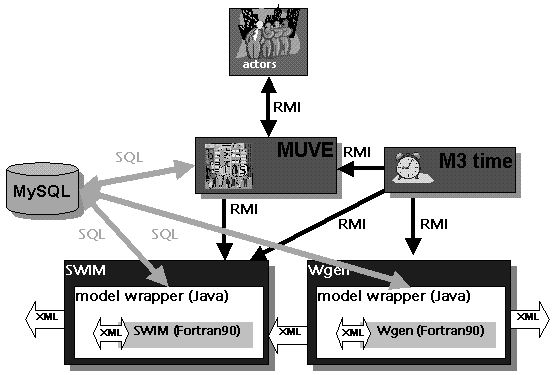

Figure 3. Part of the M3 system including two exemplary models (SWIM and Wgen) that communicate with each other via XML. A MySQL database is used for storing and accessing the data. The synchronization of the models is given by a virtual clock that provides the M3 time. The MUVE is used for the logical representation of all simulated objects

Figure 3 shows parts of the M3 system as it is implemented at the moment with two exemplary models. The simulation programs communicate with the corresponding Java wrappers via standardized interfaces. The protocol that is used to communicate with the models is described in section 3.1. The protocol makes use of a special XML syntax (http://www.w3.org/XML). Some examples of this Markup Language will be given in section 3.2. An XML filter is invoked for parsing the output data and writing it into an SQL database (MySQL) anywhere in the Internet, from where it can be retrieved using a simple web browser. The data is represented intuitively in the virtual world or by a GIS system with access to the SQL database. A model coordinator that is part of the MUVE is used for controlling the simulation models (section 4).

3.1 Interfaces and Protocols

The models that are connected to the M3 simulation use an XML data format for the communication with the other components of the M3 framework (e.g. user interfaces, other models, databases). The structure of the input/output XML data of each model is clearly defined by a Document Type Definition (DTD) or an XML Schema that serves as an interface definition of the model. The XML data of a simulation model may also include method calls. With this standardized and self-describing model interface, it is easy to encapsulate a model (e.g. For-

tran/C code) using a generic Java wrapper that allows remote access to the functionality (via RMI) and the data (via sockets) of the simulation program, including interactive control and management. The Java wrapper uses standard input (stdin) and standard output (stdout) of the simulation program as interfaces for the communication with legacy code that is not written in Java. A substantial advantage of this type of interface is that almost every programming language implements the possibility to write to stdout and to read from stdin. Single model components that use stdout/stdin for their controlling and data transfer can very easily be tested and validated by means of scripts or by simply invoking method calls at the command line.

The M3 system itself uses the remote method invocation (RMI) for the communication with the model wrapper that implements a Java remote interface.

3.2 XML Data Model

Using an XML data model for coupling models has several advantages:

- Extensibility: the semantics of the XML data model can be customized individually for each application
- Structure: the structure of the XML data is defined in an external Document Type Definition (DTD) or XML Schema. This enables XML parsers to validate and check the well-formedness of the XML data
- Text format: XML supports all popular encodings (ASCII, ISO8859-1, unicode, etc.) and is independent of the platform, the programming language, and the protocol that is used to transfer the XML data
- Documentation: the XML data format contains meta data that offers a comprehensive documentation of the inputs and outputs of the component. This meta data contains — besides format specifications (double, float, integer, string etc.) — the (e.g. physical) meaning of the data that makes the data easily understandable to humans or computer programs
- Widely-used: XML is very popular and there are many tools for editing, parsing and evaluating XML data.

The major disadvantages using XML are the large data volume that is caused by the additional meta data and by using a text-based format for encoding binary data, and the time overhead that is needed for parsing the XML data. The large data volume can simply be reduced by online compressing techniques or by transferring binary data separated from the format description over an own communication channel.

When two model components speak the same XML syntax, the simplest way to couple them is just to pipe the stdout of the one model into the stdin of the other.

An example of the XML input and the corresponding output of the simulation program SWIM (see section 2) is shown below. If the physical units are not specified, we use SI units as default, i.e. the precipitation in the example below is measured in m instead of mm.

SWIM input:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE model SYSTEM "swim1_5.dtd">
<model name="swim">
  <methodCall name="step" />
  <methodCall name="getProperty"
              propertyName="precipitation" />
  <methodCall name="setProperty"
              propertyName="interactiveManagement"
              type="boolean"
              igis="1519">true</methodCall>
  <methodCall name="thisIsNotAValidMethod" />
</model>
```

Corresponding SWIM output:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE model SYSTEM "swim1_5.dtd">
<!-- data output from swim -->
<!-- working directory: modelle/br2/ -->
<model name="swim">
 <version>1.5_m3</version>
 <methodResponse nr="1" name="step">
  <dataIO>
   <timeInstant>
    <dateTime.yearDayMs>
     <year>2000</year>
     <dayOfYear>1</dayOfYear>
    </dateTime.yearDayMs>
   </timeInstant>
  </dataIO>
 </methodResponse>
 <methodResponse nr="2" name="getProperty">
  <dataIO>
   <timeInstant>
    <dateTime.yearDayMs>
     <year>2000</year>
     <dayOfYear>1</dayOfYear>
    </dateTime.yearDayMs>
    <data.raster rasterName="subbasins"
                 rasterIndex="isub">
    <precipitation isub="1">0.70E-02</precipitation>
    <precipitation isub="2">0.10E-01</precipitation>
    <precipitation isub="3">0.96E-02</precipitation>
    <precipitation isub="4">0.10E-01</precipitation>
    <precipitation isub="5">0.11E-01</precipitation>
    <precipitation isub="6">0.93E-02</precipitation>
    <precipitation isub="7">0.92E-02</precipitation>
    <precipitation isub="8">0.74E-02</precipitation>
    <precipitation isub="9">0.74E-02</precipitation>
```

```
    <precipitation isub="10">0.10E-01</precipitation>
    <precipitation isub="11">0.10E-01</precipitation>
    <precipitation isub="12">0.93E-02</precipitation>
    <precipitation isub="13">0.10E-01</precipitation>
    <precipitation isub="14">0.89E-02</precipitation>
    <precipitation isub="15">0.11E-01</precipitation>
    <precipitation isub="16">0.11E-01</precipitation>
    <precipitation isub="17">0.11E-01</precipitation>
    <precipitation isub="18">0.11E-01</precipitation>
    <precipitation isub="19">0.78E-02</precipitation>
    <precipitation isub="20">0.78E-02</precipitation>
    </data.raster>
   </timeInstant>
  </dataIO>
 </methodResponse>
 <methodResponse nr="3" name="setProperty">
 </methodResponse>
 <methodResponse nr="4" name="thisIsNotAValidMethod">
  <fault>
   <faultCode>10</faultCode>
   <faultString>
     error: lookup for unknown function
   </faultString>
   <faultActor>
     m3.model.swim.xmlparse_expat.c
   </faultActor>
   <detail>thisIsNotAValidMethod</detail>
  </fault>
 </methodResponse>
</model>
```

3.3 XML in Fortran

In environmental simulation the majority of the programs is still written in Fortran77 or Fortran90. Due to the lack of XML parsers for the Fortran programming language we use the C library *Expat* (http://sourceforge.net/projects/expat/) which provides a Simple API for XML (SAX) for the evaluation of the incoming XML stream. Expat can be linked to the Fortran program in order to access the data and functionality of the simulation program via XML.

4 Model Control

The major challenge when coupling several simulation models is to grasp and to define the conditions and dependencies between the models. An accurate description of these relations is needed to be able to control a simulation system that incorporates coupled models.

A very suitable tool for the description and modeling of discrete systems are Petri Nets. Petri Nets consist of places ◯, transitions □, arcs pointing from places to transitions ◯ → □ (place is called input place), arcs pointing from transitions to places □ → ◯ (place is called output place) and markings •. There are two simple rules that describe the movement of markings in a Petri Net: A transition is called active, when all its input places and none of its output places has markings. An active transition may switch by removing all markings from the input places and by putting one marking on every output place. For a mathematical definition of Petri Nets refer to e.g. Reisig [1991].

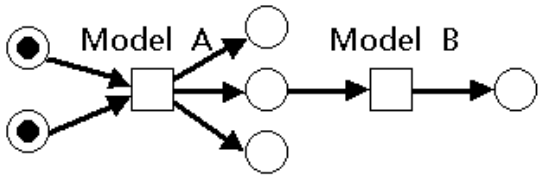

Figure 4. A Place Transition Petri Net for modeling and controlling the data flow between model components. Model A is active. When model A switches it retrieves the input data from the input places and fills the output places with output data. After that, model B will be active.

In our case, we use the transitions as a model for software components (e.g. simulation programs), the places represent files or data buffers and the markings symbolize the data that has to be transferred between the software components (Figure 4). Once the Petri Net for a coupled simulation is defined, the model coordinator just has to activate every simulation component that has all inputs places filled with markings (data) and that has all output places free to handle the output data. The Petri Net itself may be defined in terms of a scripting language or by a graphical user interface (Jüngel et al. [2000]). With a small set of predefined transitions, it is possible to implement loops or other conditional dependencies between the components.

5 Conclusions

The M3 simulation distinguishes itself from conventional simulation systems by the involvement of real human actors into the simulation. But like in other complex simulation systems, we have to deal with the challenge of coupling and integrating many different simulation models. We decided to face this challenge by using a loose coupling scheme that uses asynchronous communication on a simulation network where the individual simulation programs are distributed over concurrent processes. In order to simplify the coupling mechanisms between the models we implemented a generic, self-describing XML data format for transferring data and for invoking method calls on remote simulation models. Petri Nets are used as a powerful tool for defining the dependencies between simulation components and for the control of the data flow in a system of

coupled models.

For further information about the M3 project an the current state please contact our project web site http://mmm.first.fraunhofer.de

Acknowledgements

I would like to thank my colleagues from Fraunhofer FIRST that contributed to the M3 project, especially Prof. A. Sydow, S. Unger, H. Rosé, T. Aßelmeyer-Maluga, P. Frank, B. Walter, B. Kwella, M. Schmidt, M. L. Jugel and all the students involved in the project. From the Potsdam Institute for Climate Impact Research I would like to thank the Hydrology and Water Resources working group for providing the model SWIM, the Forests and Forestry working group for the 4C model and the Data & Computation department for the fruitful discussions, particularly C. Ionescu for the idea with the petri nets and the theory of the typed data transfer. Last but not least I would like to thank my wife Tanja for proofreading this article.

References

E. Beger, D. Bieninda, and S. Gester. Konzeption des Moduls Impaktmodelle im M3-System des Projekts GLOBALSIM. Technical report, IBB Ingenieurbüro Beger für Umweltanalyse und Forschung, Rossendorf, 2001.

H. Bugmann, R. Grote, P. Lasch, M. Lindner, and F. Suckow. A new forest gap model to study the effects of environmental change on forest structure and functioning. In G.M.J Mohren and K. Kramer, editors, *Global Change Impacts on Tree Physiology and Forest Ecosystems*, volume 52 of *Forestry Sciences*, pages 255–261. Kluwer Academic Publishers, 1997.

M. L. Jugel. Enhancing MUVEs: Connecting virtual objects with environmental simulation. In *Proceedings of the International Conference on Virtual Worlds and Simulation*, Phoenix, Arizona, January 2001.

M. Jüngel, E. Kindler, and M. Weber. The petri net markup language. Workshop AWPN, Koblenz, Germany, 2000.

V. Krysanova, D.-I. Müller-Wohlfeil, and A. Becker. Development and test of a spatially distributed hydrological/water quality model for mesoscale watersheds. *Ecological Modelling*, 106:261–289, 1998.

P. Mieth, S. Unger, and M.L. Jugel. An environment simulation and monitoring system for urban areas. *TRANSACTIONS of SCS*, 15:115–121, 1998.

W. Reisig. *Petrinetze — Eine Einführung*. Springer-Verlag, 1991.

C.W. Richardson and D.A. Wright. *WGEN: A model for generating daily weather variables*. Agricultural Research Service, U. S. Department of Agriculture, 1984.

H. Rosé. M3-Project — Actor based simulation in virtual worlds. In *Proceedings of the Biennial meeting of the International Environmental Modelling and Software Society*, Lugano, Switzerland, June 2002.

M. Schmidt, R.-P. Schäfer, and K. Nökel. SIMTRAP: Simulation of traffic-induced air pollution. *TRANSACTIONS of SCS*, 15:122–132, 1998.

S. Unger, I. Gerharz, P. Mieth, and S. Wottrich. HITERM — high-performance computing for technological risk management. *TRANSACTIONS of SCS*, 15:109–114, 1998.