Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

NEXT GENERATION SIMULATION ENVIRONMENTS FOUNDED ON OPEN SOURCE SOFTWARE AND XML-BASED STANDARD INTERFACES

Thomas Wiedemann

Friedrich-List-Platz 1
HTW Dresden
Dresden 01069, GERMANY

ABSTRACT

During the Winter Simulation Conference 2001 the OpenSML-project was presented and started. The OpenSML-project is based on the Simulation Modeling Language (SML™) and is an open source, web-based, multi-language simulation development project guided by a consortium of industrial, academic and government simulation consultants, practitioners and developers. For the simulation community, the open source movement represents an opportunity to improve the quality of common core simulation functions, improve the potential for creating reusable modeling components from those core functions, and improve the ability to merge those components using XML, HLA and other simulation community standards. This paper extends the OpenSML-project by using universal, language independent XML-descriptions and code generators for converting OpenSML-models to programs in Java, VisualBasic or C++. This would be the first time a simulation model could be transferred between different platforms without manual changes.

1 INTRODUCTION

In the last ten years simulation methods were successfully introduced in nearly all areas of science and business (Wiedewitsch and Heusmann 1995, Kuljis and Paul 2000). The main algorithms and mathematical foundations are well defined and efficient. But the real application of simulation systems is still difficult and does not reach more than 10% of all industrial firms by a number of reasons:

- Unlike the continuous simulation marketplace there is no leading discrete simulation system. The market shares of the main tools (AutoMOD, TAYLOR ED, Arena, SLX) are very different in the global regions and industrial branches. As a result, there is no universal standard for discrete simulation. Models created with he main simulation tools can not exchanged between the systems.
- As a result of the small market the prices of the systems are very high. Typical prices of up to $50,000 are too high for medium-sized firms.
- Especially in the area of optimization with simulation models, there exists a performance problem. It seems like a paradox, that an older simulation languages GPSS is significantly faster than modern simulation systems.

These problems indicate the of a need of a new strategy for the development of simulation tools. Like in the database software domain, **we need common accepted standards for modeling and simulation.** A first step could be the application of the Open-Source-idea, which was very effective and successful in the LINUX-development. If this idea would be successfully adapted to simulation software, the market for simulation tools and projects could be substantially larger. As a result there would be enough work for all simulation experts without any need of the traditional competitive thinking.

2 OPEN SOURCE - A NEW OPTION FOR DEVELOPMENT OF SIMULATION TOOLS

The main ideas of Open Source and the advantages for the simulation area are discussed in detail by Kilgore in the original paper outlining the OpenSML-project during the Winter Simulation Conference 2001 (Kilgore 2001). The most important facts are extracted from this paper include:

- The open source movement is a revolutionary perspective on how software should be created. While the movement is most often associated with the Linux operating system (Linux 2002), there are open-source initiatives throughout the software industry and new projects are constantly emerging.
- There are already simulations projects in the Open Source area. But no common and powerful simulation standards will be developed by these projects.

- Open source projects are a very fragile and sensitive network of developers, deeply interested beta-testers and end-users. In result of the non-commercial aspect of the project there must be a high degree of common understanding and social interactions.

There are many more details about the culture and the opportunities necessary for successful Open-source projects see (Raymond 1999). The main question of Open Source is the copyright of the developed tools and simulation models. In general, all Open-source code is free. But the license agreements state, that code, which is developed by using open source code must be also free of copyright restrictions. This would be a disadvantage for the simulation community, because any model developed by a free open source tool must be free too.

A good solution was found in (Kilgore 2001): "The issue of what to share and how to share it is an important issue for distinguishing between free SML simulation code and proprietary modeling code. The present plan for SML is to distribute simulation language source code under a modified Lesser/Library General Public License (Free Software Foundation, 2001) that ends where the SML simulation language ends and the SML-based simulation model starts. Normally, all extensions and modifications of LGPL licensed software must be distributed under the same LGPL license under which the software was acquired. Obviously, this restriction cannot be applied to software that uses the SML code to create a specific model. The LGPL required that the user share improvements by returning the revised code to the SML repository. Proper SML sharing principles would require that the user comply by depositing a generic or example version of the method that does not contain proprietary property names or ranking rules.

The economic vision behind SML will be to allow users to compete in a marketplace of models and modeling components based on SML, but to cooperate in the development of compatible, extendible SML objects and methods which support those models. The resulting "coopetition" among simulation companies and professionals will be better for the long term viability and profitability of the simulation industry than the current system of incompatible products and lack of standards."

This business model is similar to the industrial cooperation that allows automotive companies to standardize on lug nuts but compete on car models. hardware. It is an imperfect and delicate alliance amongst opponents that is viewed in advance as wishful folly and viewed in retrospect as insightful wisdom. Sometimes coopetition happens because governments decree that change should occur, but more often coopetition happens because influential and powerful users decree that the change should occur.

3 THE SML – LANGUAGE

Although open source development means that all specifications are developed by the whole development team, Kilgore gave some main starting points in (Kilgore 2001): "The mission of SML is to produce reusable simulation software at both the simulation source code and modeling source code levels. Reusability requires at a minimum that the code be **readable**, **modular** and **extendible**. Contrary to most simulation products, SML will sacrifice performance to achieve reusability. "

Readability means that the target audience for the code is the closer to the first-time reader with limited programming background than to the experienced hacker.

Modularity is related to readability in that a part time developer can make a change to the source code or replace an entire SML module without having to understand or modify large amounts of SML source code.

Extendibility means that SML is designed to be easily modified and repackaged for specific applications. As mentioned previously, simulation languages are usually biased towards a particular target application based on the experiences and anticipated needs of the modeler or developer. The best method of achieving the combination of readable, modular and extendible that SML desires is through object-oriented development. Even though the original SML prototype is Java-based, any object-oriented language is a potential candidate for an SML community (Kilgore, Healy, and Kleindorfer 1998).

The main OpenSML-architecture is shown in Figure 1. The actual state of the OpenSML-project is available at (SML 2002). The discussion of the common process oriented language is still in progress. It could be understand as a summary of all basic objects and functions in the different SML-implementation languages without any language specific details. The existence of such a common language is an important fact for development of automated code transformation tools in this paper.

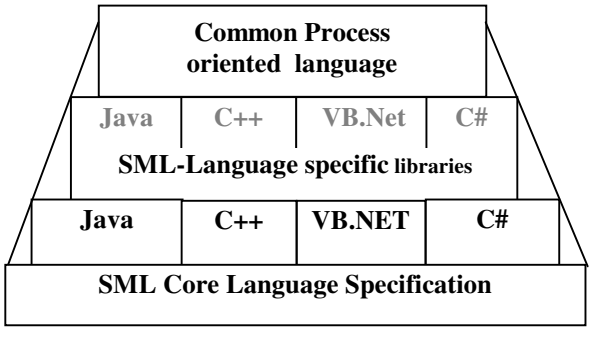

Figure 1: The OpenSML-architecture by (Kilgore 2001)

4 THE MAIN SML-CONTROL AND DATA-ARCHITECTUR

The main goal of SML was a common and universal environment for simulation and related services like optimization and animation. Therefore it needs a common and standardized interface for controlling and accessing all submodules of the system. The hierarchy of this interface is shown in Figure 2.

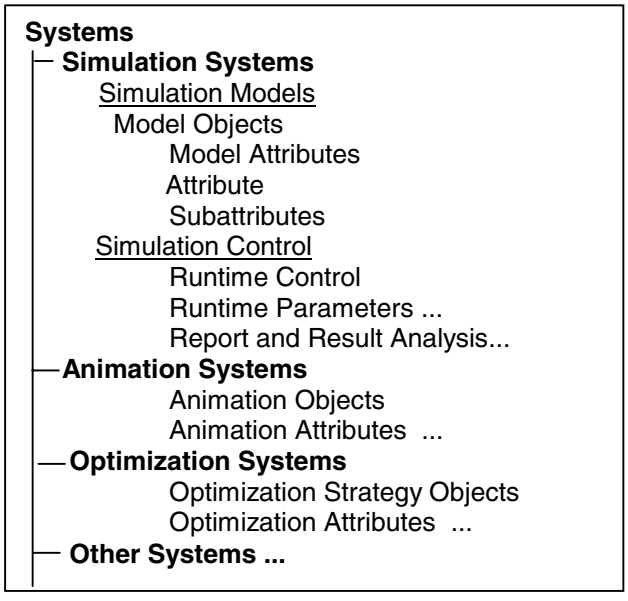

Figure 2: A Universal Object Hierarchy for Modeling and Optimization Tools

This interface allows different access-schemes and interfaces for all SML-modules. Depending on the used language each node of this object model can be addressed by a full reference

```
APP("Simulation").system("SILK").model("Testm
odel").object("Machine1")
```

or by shorter references :

```
set m= APP("Sim").system("SILK"). model("M1")

m.object("Machine1").att("Capacity")=120

m.object("Machine1").att("Worktime")=12.0
```

The advantage of full reference is that a module can be addressed any time and on any system. The disadvantage is in performance as this creates slower and more expansive programming representations. A common access standard will be defined to accommodate of the most important languages like Java,C++,C# and VisualBasic.

5 XML AS A SML-INTERCHANGE FORMAT

One main task is the definition of the SML base syntax. A attempt to define the "SML-language" as something unique to Java or C++ language would decrease the flexibility and the acceptance between all developers. As a result of this situation, it would be beneficial to define SML independent from any known language. At the present time XML seems to be the best choice for this task (Phillips 2000). A initial prototype SML-model coded with XML-statements is shown in Figure 3.

```
public void process( )
{  // SML-JAVA-Model
create( expA.getValue());
qadd( queEntity );
waituntil(Server1.isIdle(this));

qremove( queEntity );

seize(Server1 ); // get resource

delay(expService.getValue());

release(Server1);// free

if (defects.getvalue<0,05)

{ seize(Repair );
delay(Repair.getValue());

release(Repair); }

dispose( ); }
```

```
<?xml ...> ....
<process name="Process1"
  repeatinterval=" expA" >
<use name="queEntity"  type="Q"/>
 <func name="waituntil"
par1="Server1.isIdle()" />
 <leave name="queEntity "
type="Q">
 <useentity name="Server1"
type="Server" />
 <func name="delay"
par1="expService()"  />
 <leaveentity name="Server1 "
type=" Server ">
<if  matchcondition= "defects()
>0.05"><match>  <useentity
name=" Repair"  type="Server" />
  <func name="delay"  par1="exp
Repair ()"  />
  <leaveentity name=" Repair "
type=" Server "></match> </if>
<func name="dispose" /></process>
```

Figure 3: The Conversion of SML.JAVA to SML.XML

One difficult aspect to the conversion are descriptions of dynamic actions and events. Tests with full XML-encoded descriptions were long and hard to understand. A better mix of XML-structures and traditional operators is shown in Figure 4.

```
<object  name="Product11">
      <attribute name="Color">
      <attribute name=" Priority ">
 <process>
<actions>
 <event  name="OnInit">
   set  m= APP("Simulation").system("SILK").model("Testmodel")
   m.object("Product11").att("Priority")=10
 </event>
 <event name="DuringSimulation">
             generate 1 object("ThisObject") every 120 seconds
             m.object("Buffer1").store("ThisProduct")
            m.object("Machine1").workon("ThisProduct").fortime(10)
 </event>
 </actions>
  <options  output="tracelist"  animation="ON" >
 </process><object>
```

Figure 4: A Simplified XML-Format for Events and Actions

6 THE DATA FLOW AND SUBMODULES

The data flow of a SML-based simulation environment with two different simulation languages is shown in Figure 5. Above the SML-level will be the GUI-interfaces or interfaces to other information systems. The large block in the center of the system controls all processes. It is also a interfacing layer between the very specific tools at the tool level and the universal and standardized modules at the SML level.

The communication between all modules is based on file or network techniques. The communication protocol uses XML-coded information. In many cases the content of the XML-databases or XML-encoded simulation results is only wrapped by an additional XML-layer and transported over the network. Larger amount of data, for example simulation results, will be compressed by well-known compression algorithms for better transportation speed. For the end user this data conversions will be transparent.

7 AUTOMATIC TRANSFORMATION OF SML-LANGUAGES

One main requirement for the SML-system was independence from specific implementation languages. In the first draft of the SML-system this independence was seen as a result of a common semantic scheme of all SML-dialects in C++, C#, Java and VisualBasic. The same semantic scheme allows the user a conversion from one SML-dialect to an other programming language. The first draft of OpenSML allows such conversions, but does not support this idea with tools.

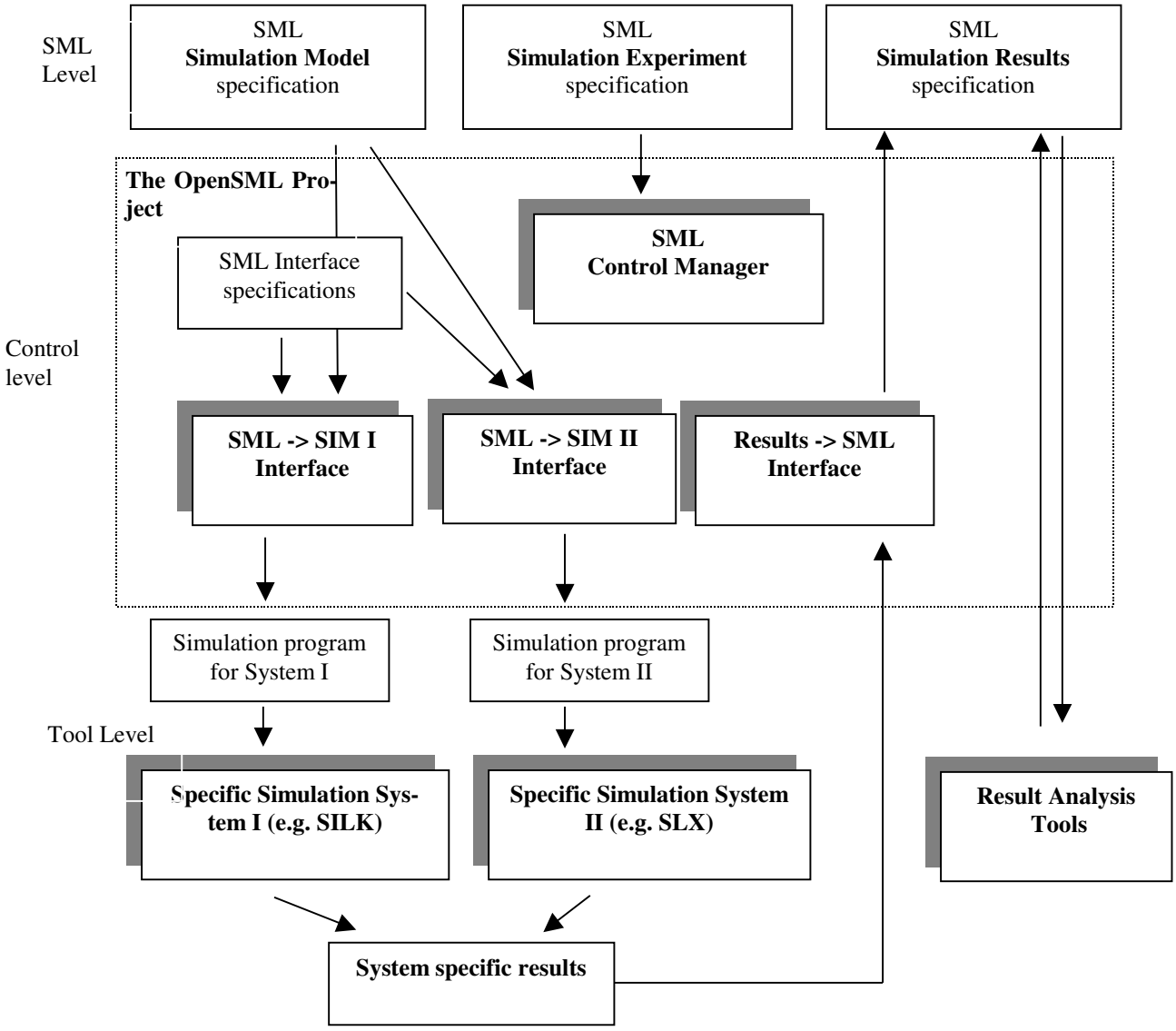

Figure 5: The Main Architecture of the SML-System

With the increasing power of OpenSML new and faster SML-implementations will be available. At that stage of development, the availability of support for code conversion will be very useful. For providing a practical solution, a fully automated code conversion program is currently under development. The module is controlled by XML-code-templates and can convert SML-programs to and from SML-XML (see Figure 6). The module will be included in the main control module of OpenSML (see Figure 5). The implementation of the transformation module is not difficult as a result of the common semantic of all SML-dialects and very similar syntax-structures. A similar code transformation module was already presented in detail in (Wiedemann 2000) .

The transformation of a model in a specific SML-language to XML is done by a code parser. This code parser is initialized also by XML-statements and generates the corresponding list of XML-statements for the SML-code.

A Java-SML-code like qremove(queEntity); is stripped from any syntax specific characters like parentheses and separators and is converted to a standardized XML-syntax <leave name="queEntity " type="Q"> . This syntax is used for all object related code.

The model is represented inside the code parser as a tree, with additional links between the list elements. The user interface of the transformation module shows this tree and allows changes on the data.

The transformation from a XML-model back to a SML-language is similar. Although this task can be solved by freely available XSL-filters, the code generator is a specifically developed software. The main goal is a consistency between the control information for the code-parser and the code-generator. With only one definition of a keyword in the XML-Codetemplates, like "qremove", both directions of the code transformation are controlled. This allows for better management of the control data and keeps changes at only one place.

Currently the XML-tools are based on the SAX-API, which is faster and smaller than the DOM-API. During the parsing process all SAX-events are saved in an internal memory model of the parsers and code generators. Although this procedure is similar to the DOM-tools, the resulting data structures are already prepared and optimized for the code transformation process. Regarding a future usage on a internet server all tools are divided in a pure command-line module and additional user interface, which can be removed easily.

8 SUMMARY

The OpenSML open source simulation project could be a potentially beneficial evolution in the simulation software development model.

The first advantage is the larger flexibility from the use of standard commercial programming languages and development tools. Instead of having only one system the end-user can select the best solution depending on the needed interfaces and performance aspects.

The second advantage is the Open Source Lesser/Library General Public License licensing model. This license model is a good mix of the Open source principles and the requirements of simulation customers.

The third advantage is the usage of an universal, language independent XML-description, which is presented by this paper. Code parsers and generators convert OpenSML-models to programs in Java, VisualBasic or C++ and also back to XML. With two sequential transformation processes a simulation model can be transferred between different platforms without manual changes. Together with the basic SML-idea a universal and non-language-dependent system can be provided in the near future.

The actual state of the OpenSML-project is ongoing and further information is available at (SML 2002). Its future development will provide the first time in simulation history a universal and open simulation system. Any interested simulation expert or user is invited by the authors for

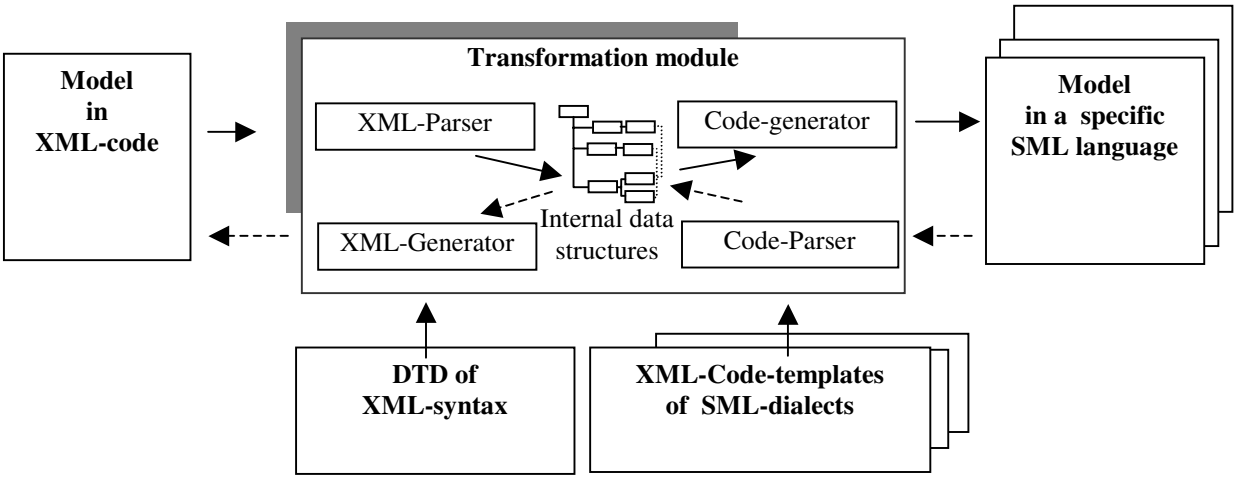

Figure 6: Automatic Code Conversion from and to SML

sharing his ideas, experience and cooperation inside the OpenSML-consortium.

ACKNOWLEDGMENTS

I would like to thank Rich Kilgore of ThreadTec for all of his help and support in discussing the SML-project and the main ideas of this paper.

REFERENCES

Linux 2002, www.linux.org [accessed August 2002]

Kilgore, R. A. 2001. Open source simulation modeling language (SML). In Proceedings of the 2001 Winter Simulation Conference, ed., B. Peters,J. Smith. Piscataway, NJ: 2001

Kilgore, R. A., Healy, K. J. and Kleindorfer, G. B. 1998.The future of Java-based simulation. Proceedings of the 1998 Winter Simulation

Kuljis, J. and R. J. Paul, 2000: A Review of web based simulation: whiter we wander?, *Proceedings of the 2000 Winter Simulation Conference*, Orlando Florida, page 1872-1881

Phillips, L. A. 2000. Special Edition using XML. Que Bestseller Edition, 2000

Raymond, E. 1999. The Cathedral & The Bazaar. O'Reilly

SML, Simulation Modeling Language. Available online via http://www.threadtec.com/sml [accessed August 1, 2002].

Wiedemann. T., 2000. VisualSLX – an open user shell for high-performance modeling and simulation, *Proceedings of the 2000 Winter Simulation Conference*, Orlando Florida, 1865-1871

Wiedewitsch J.; and Heusmann J. 1995. "Future Directions of Modeling and Simulation in the Department of Defense", Proceedings of the SCSC'95, Ottawa, Ontario, Canada, July 34-26, 1995

AUTHOR BIOGRAPHY

THOMAS WIEDEMANN is a professor at the Department of Computer Science at the HTW Dresden. He has finished a study at the Technical University Sofia and a Ph.D. study at the Humboldt-University of Berlin. His research interests include simulation methodology, tools and environments in distributed simulation and manufacturing processes. His teaching areas include also intranet solutions and database applications. Email : <wiedem @informatik.htw-dresden.de>