Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

VISUALSLX – AN OPEN USER SHELL FOR HIGH-PERFORMANCE MODELING AND SIMULATION

Thomas Wiedemann

Technical University of Berlin
Franklin Strasse 28/29, FR 5-5
10587 Berlin, GERMANY

ABSTRACT

SLX by Wolverine software is actually one of the fastest simulation languages. Besides the high performance the SLX-compiler can be extended very easily by user specific syntax rules and new basic functions. This "pyramid power" of SLX is used to build a new system for modeling and simulation – VisualSLX. This system is a shell atop the standard SLX-compiler and the runtime system. All model and simulation data are stored in a universal database. VisualSLX could be used for a comfortable, rapid visual modeling and for remote modeling and simulation through the internet without any knowledge of the SLX-syntax and modeling paradigms. This paper reveals the architecture and the underlying data structures of the system. Additional requirements and interfaces are caused by the application of VisalSLX as a web-based modeling and simulation system.

1 INTRODUCTION

In the last ten years simulation methods were successful introduced in almost all areas of science and business (Wiedewitsch and Heusmann 1995, Roberts and Dessouky 1998). The main algorithms and mathematical basics are well defined and efficient.

At the moment the focus of research is shifted from the kernel simulation functions to more peripheral tasks like graphical modeling, 3D animation in the Web with VRML, intelligent result analysis with Data-Mining tools and optimization methods.

However especially in the area of optimization with simulation models we see a revival of an old critical factor - the performance of the simulation system. It seems to be a paradox that almost 35 years old simulation languages like GPSS are significantly faster than modern simulation languages and systems.

2 SIMULATION PERFORMANCE

At first glance simulation performance is considered to be solved by the help of Moore's law and his physical realization in Giga-Hertz processors. If only one experiment with one simulation run is taken into account, there is no practical difference between 2 seconds or 1 minute of one run. But if the same model is integrated in a optimization cycle with one thousand experiments and 20 runs per experiment it needs 46 minutes in the first case and nearly 14 hours in the second case. Thus with the increasing usage of optimization methods in simulation the performance of the simulation system becomes one of the most critical parameters. The performance of simulation systems is influenced by the following factors:

- The most critical performance killers in modern systems are all visualization and reporting functions during run-time. It can be shown very easily, that even small and inexpensive 2D-bitmap animations and parameter indicators consume about 90% of the processor power. 3D-animations even intensify the problem. Modern 3D-graphic adapters with hardware support for animations could defuse the situation, but the necessary amount of information interchange to move an object in the virtual space will sometimes exceed the amount for the abstract movement inside the simulation model.
- A lot of modern simulators like TAYLOR II contain proprietary script languages for defining application specific strategies for scheduling and routing. Regrettably there are no available details about implementation of this script languages, however the scripts seem to be interpreted or executed in an interpreter-like mode. It is almost well known from the roots of computer science that besides a lot of advantages the performance is the weak side of interpreters. Therefore their

usage in simulation tools is very critical under performance constraints.

- The performance of older simulation systems, like GPSS/H is often better than the performance of modern object oriented simulation systems. One reason for this very astonishing fact is the immoderate and sometimes needless usage of inheritance techniques. Especially if "virtual" class methods are used, for each call of an object method the processor must fetch the object data pointer, then it has to calculate an indexed pointer to the table of virtual functions and then it receives the correct address of the function. Compared to one single call of a function this procedure needs 3-20 times more processor cycles. The highly sophisticated cache and queuing buffers of the modern processors could be knocked out by this procedure and in result the processor falls back to the single speed of the main memory.

As a result of the discussed performance problems, the applicability for optimization of actual simulators differs widely. So the paradox situation has occurred, that for very expansive and complex simulation experiments with optimization tasks GPSS is still used by large companies. The historical merits of GPSS and its performance are well accepted, but the main syntax and semantics are still at the level of 1964. Therefore it was registered with interest, when Wolverine Software, the vendor of GPSS/H released a successor for GPSS – the SLX –system.

3 THE "PYRAMID POWER" OF SLX

In difference to GPSS the new SLX system uses a modern C-like syntax. The source text is well structured and supports subprograms and include files.

The first important feature of SLX is its performance. As mentioned in (Henriksen 1995) this is achieved by very efficient list structures for the future event list and the generation of native assembler code by the SLX compiler. SLX is probably still one of the last modern simulation languages which optimizes the generated machine code at the assembler level. The results are impressive – SLX is 3 to 20 times faster than other known simulation tools like TAYLOR or SIMPLE++.

The second important feature of SLX is its extensibility. The compiler itself can be extended at compile time with new syntax rules and commands. The power of this feature outstrips all known C++-macro definitions. An excellent example is the H5-extension, which makes SLX understanding GPSS programs (Henriksen 1999).

Although the performance and extensibility of SLX are excellent, there are also some deficits:

- Due to the traditionally text based user interface the readability of a SLX-model is lower than models in Arena or Taylor. This is especially true for inexperienced modelers.
- The SLX syntax is more powerful than GPSS but the required code for standard model objects like buffers and machines is longer and more variable definitions are required.

In general SLX obtains very good perspectives due to its performance and flexibility. It seems to be useful to add some higher modeling level to the basic SLX system. One option - a database user interface is presented at the following pages. This system allows a very comfortable modeling similar to Taylor, but with the performance of SLX.

4 THE VISUAL SLX SYSTEM

The regularity of the SLX syntax and its well structured source code allow an automated generation of SLX programs from a high-level modeling workbench. As a result of earlier projects we decided to realize this system not as a hard coded program generator, but as an open and flexible code generator based on meta-models. The term "meta model" in this context does not denote an universal, world reference model, but a relative simple template for a class of application specific models like production planning, logistics or network models. Each specific meta-model defines:

- a set of application specific simulation objects like machines, buffers, cars or network routers;
- a set of parameters for every object with type information, default values and handling options;
- the code template for the used simulation language, in this case SLX.

The user of the modeling system defines his instances of the meta-model objects and the necessary parameters. All data is stored in the database. A model management system allows different versions of a model and stores the history of the model.

In order to simulate the defined model the user model data is combined with the code templates of the meta-model objects. The SLX program is compiled by the SLX-compiler and executed by the run-time system in command-line mode. Results are stored as text files by the SLX system. They can be imported to the database for further analysis. A second option of result analysis is given by traditional programs like Excel or future Data Mining tools (Wiedemann 1998).

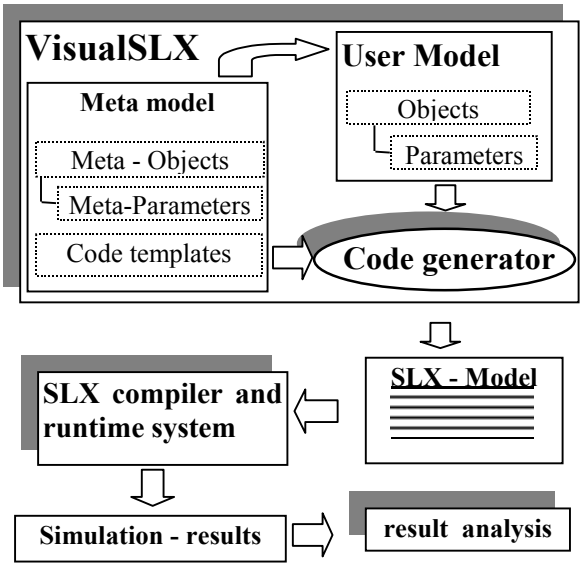

Figure 1: The Main System Architecture

4.1 The Modeling Workbench

A common user of the modeling shell uses the following operations:

- create or select model
- create, select or copy version
- create or select model object
- edit object parameters.
- code generation and start of the simulation result analysis.

A first working application of this modeling workbench was realized with Access. This approach was used for rapid prototyping the data structures and the necessary algorithms. The final VisualSLX Workbench will be realized as a web-based interface (see details of implementation).

In the first Access-Workbench all user operations are simply done by editing the database entries. The user does not need any knowledge about the SLX system or the related meta-model. The data of the meta-model are used for defining object types in the user model. Almost all labels and pull-down-lists in the model instances are generated from the related meta-model definitions. During a model edit session the meta-data definition can be changed by the administrator of the VisualSLX-shell. This is especially useful in the early state of a meta-model, when the meta-model itself is created and tested.

The main form for defining a model of the VisualSLX user shell is shown in fig. 2. The center of the screen show all existing model objects and the type of the object. The subform below shows the parameters of the selected object. The number of parameters is unlimited and depends only on the used meta-model.

Each parameter field can comprise not only a single number but also complex definitions like functions calls (e.g. rn_uniform(Service,10,30) which calculates a random value in SLX). If the user shell is used for other simulation systems, such complex definitions must be transformed from a common representation into the language specific syntax. Otherwise the model will be language specific.

4.2 Details of Code Generation

The templates for code generation are stored as text fields in the database. The length of each database ranges from single words to some lines (see fig. 3).

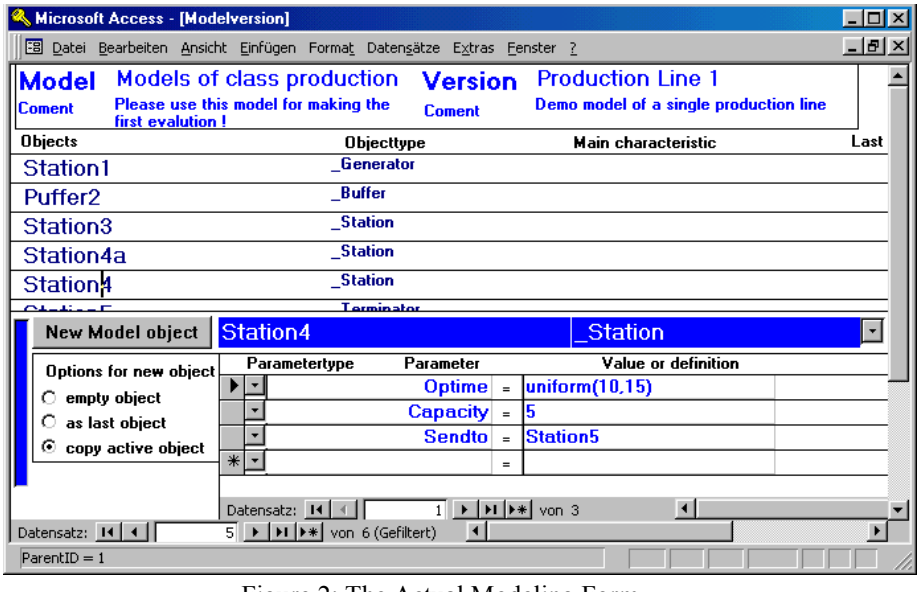

Figure 2: The Actual Modeling Form

Figure 3: Code Templates

All fields for user data must be marked with variable names and special symbols like # as markers. So the operation time for a machine object is defined as

```
advance  #optime#
```

where "advance" is the regular command of SLX for delaying the moving object and "#optime#" is the parameter which is taken the user model during code generation.

In result of the well structured SLX syntax the code generator writes code to different sections of the SLX source code file.

In queuing models typical sections are:

- global declaration area for variable and objects
- local declaration area for local variables
- main process model
- control structures.

The target section of each template line is specified in the database entry of the code generator. The number and names of sections are not limited and are defined also in the meta-model.

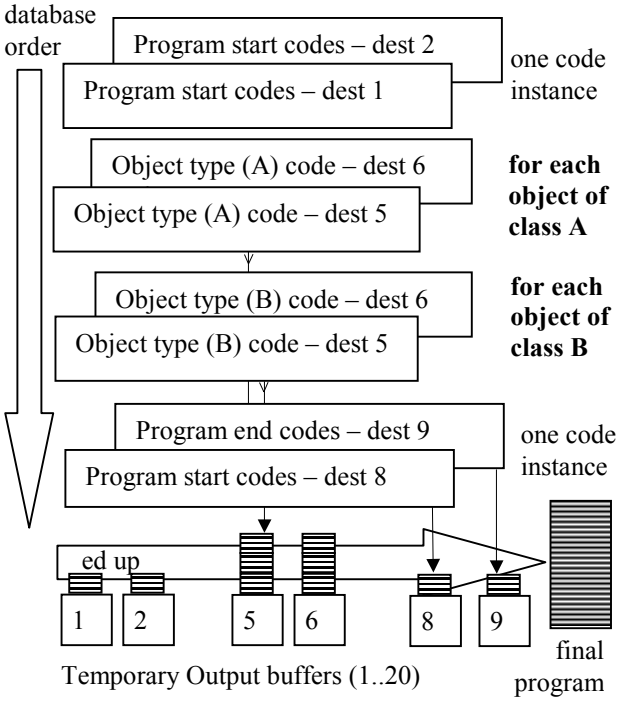

Figure 4: The Code Generation Process

4.3 Simulation processing

The generated source file is stored in a local directory and SLX is called from the user shell in command line mode with options for output of all important data to files. Both, the local directory and the command line options are stored in database tables for fast adjustments of the VisualSLX-System.

The simulation system is started from the user interface with the shell-command of the operating system.

In case of syntax errors in the source code of the model (which is possible, if a new code template is tested), the SLX compiler returns a error listing. In any case the control returns back to the user interface.

When SLX is started as minimized icon the user does not recognize this external procedure, but works only with the user shell.

In general the simulation model contains report functions for writing a trace file. The current format of the trace-file is defined as:

```
Time, Event-type, StaticObjectID, DynObjectID
 100,    1,          1,              1
 300     2,          1               1
 300     1,          2,              1
```

With this trace file each event in the model reports the actual time, e.g. the affected machine and product and the type of event (Entry , Exit , Breakdown-Start,..). The trace file is imported by the user shell after the simulation.

4.4 Model data handling

The proprietary data structures of current simulators cause tremendous problems for all users. The solution for this problem should be the same as it has been for business systems years before - a database. Thus in the presented user shell all simulation data is stored in a database, which includes:

- model data with all parameters and definitions of the model behavior,
- experiment parameters and optimization methods,
- simulation results and statistical values.

The main problem of this simulation database consists in the high complexity of real processes. The database structure has to allow a high flexibility of relationships between all objects of a simulation model. As it concerns other information systems, there is a serious discussion about the usage of relational or object-oriented databases.

Regarding that relational databases have a strong performance and well defined interfaces, we use a relational database. This should not be considered as a general decision against object-oriented databases.

The Entity-Relationship-Model (ERM) of the simulation database is shown in fig. 5. This database structure is of the same kind like as the SIMCO system, which was presented in Widemann (1998).

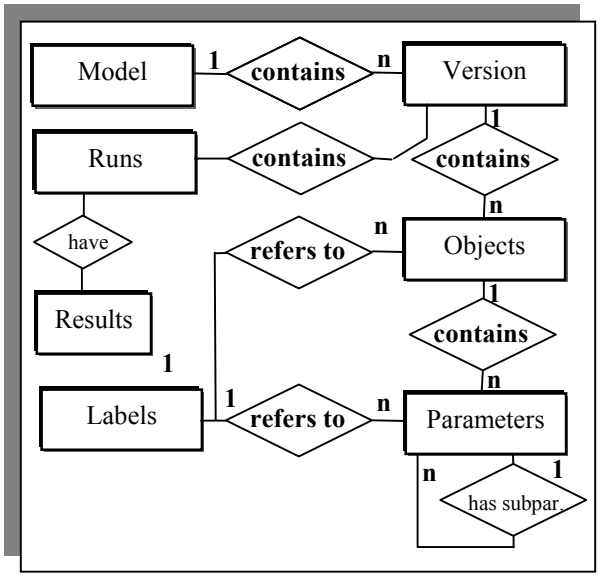

Figure 5: The ERM-Model of the Database

The attributes of the database-tables are divided into two groups:

- Attributes for administration and for building the relationships between the different levels of the model objects (objectid, parentobjectid, datatype, subtype, unique database ID).
- A set of attributes for the storage of the object data themselves. Currently we have 3 integer fields (i,j,k), 3 double fields (d,e,f) and two string attributes (s,c).

This data structure is currently used for all model entities. The most important relationship is created by the objectid-parentid relation. Each object can have N child-objects or parameters. Sets of parameters can be stored as a list of sub-parameters. There is no restriction concerning the depth of the parameter hierarchy. We are limited only by increasing access time, because each additional level requires one more memory-access.

The internal data names like i,j,k are covered by dynamic changing labels in the user interface. The definitions of the labels are stored in the meta-model.

The storage of all model and meta-model data in a database provides external applications with access to all simulation data. With SQL and a ODBC-driver a client system is able to ask for information by using SELECT-statements or is capable to make changes by applying UPDATE-commands for all model and experiment parameters in the simulation database.

5 VISUALSLX IN THE WEB

The user shell can work in three modes:

- as a traditional, stand-alone system,
- as a multi-user database in a local network,
- and as a real client server system in Intranet or Internet environments.

The first two modes are realized by the database forms shown. New or application specific forms can be developed in a short time by using latest technologies of assistant supported database design (e.g. in Microsoft Access).

The third mode requires an additional module for interfacing to the web. Due to existing powerful software components for internet applications, this interface is realized as combination of the VisualSLX code generator and a web-server (see fig. 6). A CGI-interface or similar technology does not exist. Database related requests from the web are received from the Winsock-component in the VisualSLX/WEB-application and are answered immediately. Advantages of this web-server integration are:

- a very high performance in result of direct data-exchange and open database tables in stand-by-mode,
- a long-time connection between the client and the server with continuous data flow during simulation processing or result processing .

There is almost no difference between the generation of a SLX source file for simulation and generating a HTML-form for a web based user interface ! Due this fact we can use the same code generator for HTML generation. There are no SLX or HTML specific sub programs in the code generator itself. Thus the code generator can be used for almost all text based simulation languages or text based user interfaces.

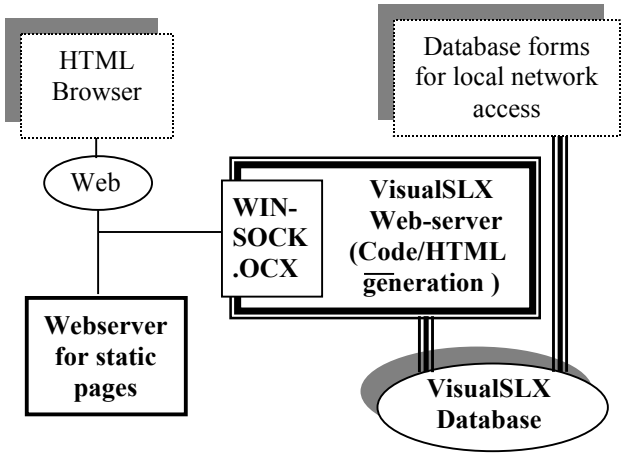

Figure 6: The Access Modes of the Shell

6 WEB SPECIFIC REQUIREMENTS

In result of specific characteristics of the actual Internet technologies some typical problems have encountered.

6.1 Web Performance

Outside an intranet the bandwidth is very often critical and is rapidly changing. For this problem we see a solution in parallelly editing more than one entry. For example all objects and their parameters could be offered in form fields at the same time. Checking operations are done with one connect to the web server and all problems are reported to the user at the same time. A second option is the usage of more than one browser window and a interleave interaction mode of the user. The best solution would be a Java-based user-interface which performs all major operations at the client side. For time reasons this solution is planned for a future version.

Due to the delay of information transfer from the server to the Web, the state of the simulation system and the visualization on the web based client user interface may differ within some seconds. So the feedback for user interactions like Break or Stopping the simulation will double the interchange time (from the client to the server and back), of up to 10 seconds. The main solution is given by the further technical improvement of the Web or a usage inside a Intranet with guaranteed quality of service. A simulation at the client side is not very useful due to the current low performance of Java.

6.2 Multi User Problems

In a multi-user local database mode a database record is locked, while a user edits the content. During this time the record is visible to other users, but can not be edited.

The Web is a system without defined sessions. Thus a user can switch off the computer or close the browser during an edit operation and the database does not receive any information about the loss of the connection. Similar to modern client server system this problem can be solved by a timeout of the lock mode. In the current database this locking operation and the check for timeout is explicitly done by VisualSLX.

6.3 License and Security Constraints

Web-based simulation also creates new requirements for software licensing and project management. Traditional software licenses of simulation packages only allow a single place usage. In general a web-based system must have a network license. The payment of the simulation customers can be done per project or by time used.

A very critical fact also consist in data security. If a company uses a web-based simulation system possibly sensible data will be stored in an external database. In order to provide a safe simulation study some secret data could be encrypted with a public key from the company. A special decrypt-DLL will be included in the import routines of the simulation model. The private key for decryption is directly transfered between the customer and the decryption module. If the source code of the decryption-DLL is validated by an external institution, the security of the private input data for simulation will be very high.

7 FUTURE DEVELOPMENT

The perspectives of SLX and the presented VisualSLX system are very interesting. Thus the further development of this concept will enclose the following activities in the near future (please look at `<http://www.aedv.cs.tuberlin.de/simco>` for actual information):

- online-animation with VRML or Flash,
- result analysis with database and data mining tools,
- integration of the SLX-HLA interface, which was developed by the Technical University of Magdeburg,
- parallel and hyper-computing with a distributed version of VisualSLX and run-time versions of SLX.

8 SUMMARY

This paper has presented a shell for the SLX simulation system. The most important features of the system are:

- a flexible and open meta model for definition of application specific classes of simulation models;
- all model, meta model and simulation data are stored in databases:
- SQL as a set-oriented language for the management of the model and simulation experiments,
- a universal code generator for the SLX source code and the HTML code for a web based user interface.
- Flexible architectures for data-interchange and bi-directional control with other, complex information systems,
- Flexible and powerful interfaces to related software packages concerning statistical result analysis, presentation, optimization, Data-mining and knowledge reasoning.

By the help of the system the needed knowledge for making a successful simulation with SLX is reduced significantly. By using the shell a user needs only common knowledge about modeling and simulation and no

programming practice in SLX. With additional user interfaces in Excel, Access or in the Web modeling with SLX will be available also for managers and decision makers.

At the first level of meta-model management a VisualSLX administrator can modify predefined examples by their used labels and parameters. This approach is especially useful for model domains with different naming covnetions but equal simulation algorithms.

At the second and highest level of meta-model management a VisualSLX administrator is able to create new meta-models. In general this requires a very good knowlegde of SLX.

The VisualSLX shell could be seen as an additional level of the SLX-pyramid (see Figure 7 and Henriksen (1999)). The new level adds a high level modeling approach to the powerful SLX base pyramid.

The VisualSLX system is currently tested and will be available as an add-on product to SLX. The usage of the system as a user shell for other text based simulation systems is possible.

Even the first prototype of the new simulation environment allows very interesting applications in the area of client-server-simulation and hyper-computing. Thus the presented concept could be an interesting perspective for the future development of modeling and simulation.

The underlying universal database structure could be seen as a first attempt to a common interchange format in the area of modeling and simulation. Together with the strong performance of SLX the presented concept could be seen as an interesting step towards a really new concept of flexibility and performance in simulation.

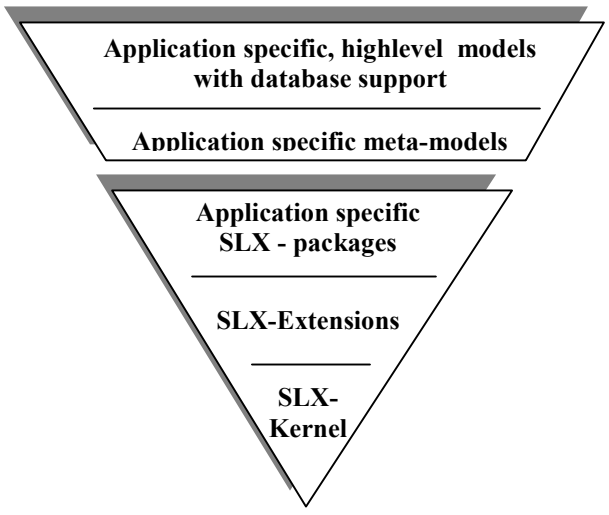

Figure 7: The Extended SLX –Pyramid (Henriksen 1999)

REFERENCES

Henriksen, J. 1995. An introduction to SLX. In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexopoulos, K. Kang, W. Lilegoon, D. Goldsman, 502-509. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Henriksen, J. 1999. SLX: Pyramid Power. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P. Farrington, H. Nembhard, D. Sturrock, G. Evans, 167-175. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Roberts C. and C. Y. Dessouky. 1998. C. An overview of object-oriented simulation. *SIMULATION* 70(6).

Wiedewitsch J. and J. Heusmann J. 1995. Future directions of modeling and simulation in the Department of Defense. In *Proceedings of the SCSC'95*, Ottawa, Ontario, Canada.

Wiedemann, T. 1997. Perspectives of component-based modeling and simulation. In *Proceedings of the Wold Congress on Systems Simulation*, Singapore.

Wiedemann. T. 1998. SIM-MINING AND SIMSQL - A DATABASE ORIENTED APPROACH FOR COMPONENT-BASED AND DISTRIBUTED SIMULATION, In *Proceeding of Summer Simulation Conference*, Reno Nevada.

AUTHOR BIOGRAPHY

DR. THOMAS WIEDEMANN is a scientific assistant at the Department of Computer Science at the Technical University of Berlin. 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.