INTERACTIVE WEB-BASED DISCRETE-EVENT SIMULATION: A MAJOR CONTRIBUTION TO BLENDED LEARNING

Wolfgang Kühn, Michael Kordt, and Roland Grah
Faculty E – Electrical, Information and Media Engineering
University of Wuppertal
Rainer-Gruenter-Str. 21, 42119 Wuppertal, Germany
E-mail: wkuehn@uni-wuppertal.de

KEYWORDS

Blended Learning, Web-based Interactive Simulation, Discrete-Event Simulation.

ABSTRACT

This paper presents an approach to blended learning in the area of production and technology management. The approach complements traditional knowledge transfer with simulation-based experience learning. Linked to a web-based learning management system, users have the possibility of supplementing theory with individually paced simulation experiments, thus gaining a profound understanding of the dynamics of complex systems.

Following the outline of a blended-learning scenario at the University of Wuppertal, the paper describes a prototype implementation of a client/server simulation application that has been successfully integrated in this year's teaching. An example of the adoption within a web-based training module sketches the numerous possibilities of individual user interaction.

INTRODUCTION: SIMULATION IN EDUCATION AND TRAINING

Modern simulation techniques are increasingly applied in the field of production and technology management in order to analyze and optimize production systems and logistic flows. The dynamics of these complex systems can be analyzed and demonstrated by use of simulation techniques regarding various demands on different user levels (Banks et al. 2005). Due to future requirements proficiency in simulation and optimization techniques is increasingly important for students and needs to be integrated into university teaching. Experience shows that interactive simulation is a suitable tool to improve the impact of teaching in the field of complex system dynamics.

The approach presented in this paper combines traditional lectures with individual e-learning and simulation-based learning units to form a blended-learning scenario, by use of online simulation (figure 1). This type of learning will be increasingly used in future, and the presence of both types of learning distinguishes blended from mere online and mere classroom methods of learning. Blended-learning strategy offers additional flexibility required to cope with future demands. In the field of teaching dynamics of complex systems the integration of discrete-event simulation is a very important step towards personalized experience learning.

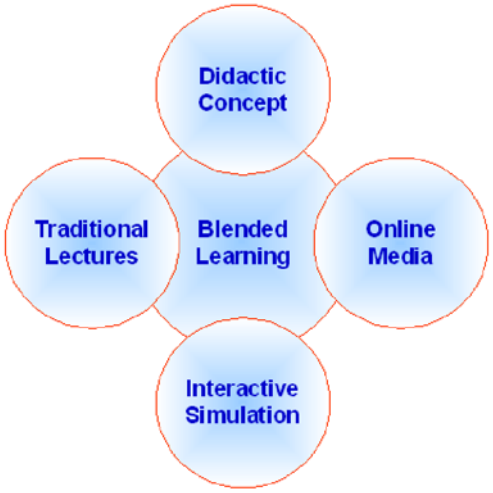

Figure 1: Simulation-Based Blended Learning

Blended-learning units applying simulation technology enhance understanding of and experience with dynamic system behaviour. Furthermore, they encourage discovery-oriented studies by allowing each student – or student group – to perform individual simulation experiments independent of location and time. Although these studies can be performed during lectures or training sessions at the university with the attendance of a coach, they will typically be part of e-learning units, with the advantage of unlimited time for experiments.

BLENDED-LEARNING SCENARIO "SYSTEM OPTIMIZATION / SIMULATION TECHNIQUES"

The project here presented is part of the initiative "e-teaching@university - best practice" and implements a blended-learning concept with simulation-based experience learning in the area of production and

Proceedings 20th European Conference on Modelling and Simulation
Wolfgang Borutzky, Alessandra Orsoni, Richard Zobel © ECMS, 2006
ISBN 0-9553018-0-7 / ISBN 0-9553018-1-5 (CD)

technology management. The goal of the project is to set up advanced blended-learning scenarios by integrating complex event-oriented simulation with a web-based learning management system. This establishes discovery-oriented learning in the area of dynamic production systems with individual feedback and learning speed.

This blended-learning concept has been set up for the first time in the master-degree course "System Optimization / Simulation Techniques" which is part of the module "Production and Technology Management" in the master studies of "Print and Media Technology". The course consists of the following learning units:

- Systems theory
- Queuing systems
- Introduction to simulation
- Simulation tools
- Simulation studies
- Simulation data
- Optimization

The interactive discrete-event simulation approach uses the potential of dynamic feedback in order to improve the impact of teaching in the fields of dynamic production systems, production logistics etc. The applied concept enables students to access simulation-based e-learning units from the laboratories inside the university as well as via internet independent of location and time.

Moreover, working in small learning groups offers an efficient learning approach and additionally imparts some knowledge in project work. Students may benefit from the discussion of results obtained from simulation experiments performed jointly.

BASIC CONCEPTS

After having sketched the general idea of complementing web-based learning units with interactive simulation-based training modules that allow for advanced experience learning, this section will outline the architectural concepts upon which the prototype of the simulation client has been built.

Figure 2 depicts the setting described in the preceding sections from the architectural point of view (Kühn 2005). The approach to blended learning applied in this project is founded upon a combination of two servers – one hosting the learning management system, the other running the simulation services. Users may access web-based contents of the entire course from a wide variety of locations by logging in at the learning management system. This in turn brings up the simulation client interface whenever a simulation-related training module is accessed.

Since ZIM – the computer centre of the university – is responsible for administering the learning management system, this paper focuses upon the design of the simulation service that runs on a different machine.

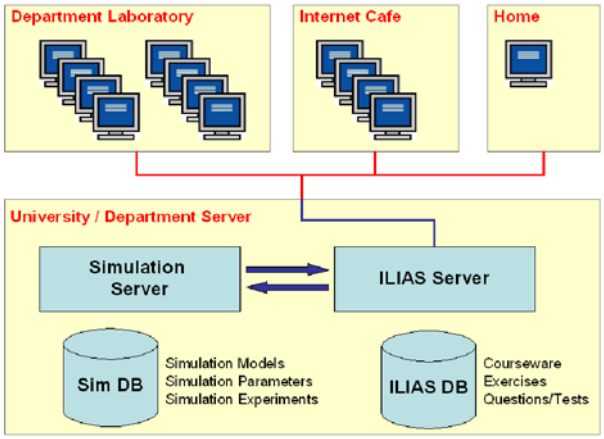

Figure 2: System Architecture

The discrete-event simulation developed so far (Kühn 2002) is not solely intended to serve this new approach to blended learning. Thus, the general architecture of the simulation system is a client/server application where the server offers, among some accompanying services, access to the discrete-event simulation, as there will be different clients that may use this simulation services for their specific purposes.

The pivotal element of the server is the simulation engine which provides a typical discrete-event simulation of production systems. Components of such systems are modeled by rather abstract elementary material flow objects like various types of workstations, buffers etc. (Becker 1991). Besides the simulation engine, the server must provide i/o interfaces that allow data of various categories (model structure and parameterization, production logs, event protocols etc.) to be saved to and retrieved from a relational database as well as XML files. With respect to its integration with a web-based e-learning environment where a growing number of students will work at individual times on different tasks involving different simulation models that may be individually parameterized and evaluated, the server will also have to take responsibility for the organization of all simulation-related data, i. e.

- predefined simulation models with initial parameterizations related to specific tasks;
- user-specific parameterizations for individual simulation experiments;
- resulting data (e. g. production logs) gained from simulation runs.

This requires some kind of user management.

The client side of the applied blended-learning scenario needs to be closely linked with the training modules that represent an important part of all course-related materials. A training module containing tasks that are concerned with a certain simulation model must enable students (typically by clicking on a link) to start up the simulation client, a graphical user interface that allows for

- examining the model structure (not editing it – here the focus is not on modelling production systems);
- reviewing and (at least partially, may depend on associated tasks) adjusting the parameterization of the components of the model;
- saving a parameterization and reloading previously saved ones;
- starting the simulation of the model, based upon the parameterization currently loaded;
- viewing and evaluating the data resulting from such a simulation run, by means of tables as well as of charts; and
- exporting selected results from such tables to third-party applications like Microsoft Excel, for the purpose of further calculations depending on the associated tasks.

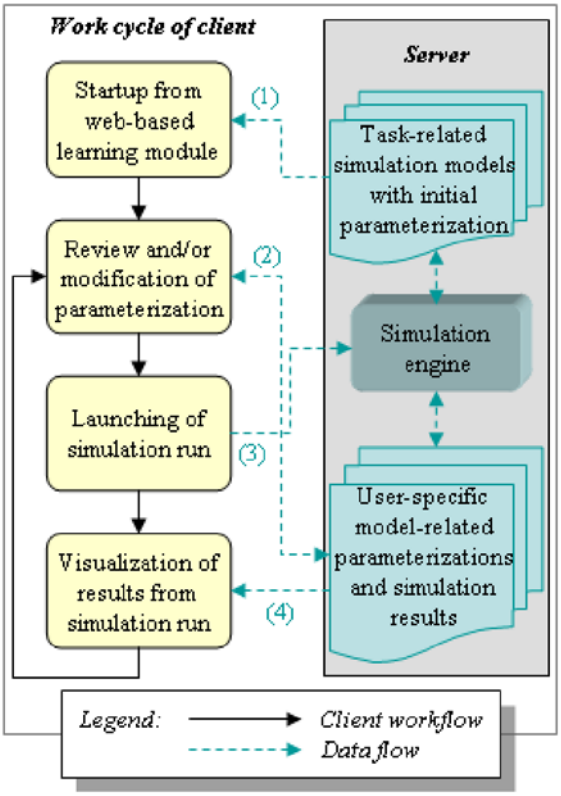

Figure 3: Client Work Cycle and Data Flows

A typical work cycle of this client is illustrated in Figure 3. When started from a given training module, the client has to load the module-specific simulation model and its initial parameterization from the server (1). It will then allow the user to review this parameterization and modify selected parameters, or to load another model-related parameterization the user has already saved during a previous session. If parameters have been modified, the user will have to save the parameterization again in order to be able to start a simulation run based on these settings (2). Upon user request, the server will launch a new simulation run (3) and, upon termination, provide the client with the results (4) which will be visualized in tables and charts. The user may then find that the results fulfil the expectations or decide to adjust some parameters again and to rerun the simulation.

A CLOSER LOOK AT THE PROTOTYPE

Although already used in this year's teaching, the application is still a prototype, which means that the implementation of some of the concepts described before is still at a preliminary stage. Nevertheless, it has proved its usability during periods of intense simulation activities in conjunction with training modules of the course "System Optimization / Simulation Techniques".

Implementing the i/o interfaces of the simulation server, attention has been paid primarily to the XML interface so far. This file-based interface is capable of reading models and related parameterizations in order to perform a simulation run as well as saving the simulation results – production and content logs (the latter especially useful for tracing a buffer's charge over time), event protocols (for debugging purposes mainly) etc. Especially with respect to the first serious application of the system during the course mentioned above the XML interface has been favoured over the database interface because XML is human readable, which simplifies testing. Furthermore, XML can be considered a cross-platform standard for data exchange, so its support may be useful for future enhancements of the system (Wiedemann 2002).

To achieve a proper storage of user-specific data that ensures a clear correlation between a simulation model, user-generated parameterizations and related simulation results, the server constructs a directory structure that depends on the user login and the model to be simulated, which itself is directly associated with a specific training module from the learning management system.

The graphical client interface – figure 4 depicts a sample screenshot – consists of several views. The first view, located at the top of the client window, provides a graphical overview of the structure of the simulation model. The model picture is generated spontaneously, evaluating the model structure retrieved from the XML

file with methods from the field of graph theory since a graphical model editor is not yet available. Below this view there is a tree representation of the model structure which reveals hierarchical groupings of components into subnets. This is useful for a comfortable evaluation of simulation runs. The tree lists all components of the model and allows for single-row or multiple-row selection, the latter again especially useful for evaluation purposes. The view on the right of the tree contains several tabs that allow for

- adjusting the parameterization of a component selected in the tree view (tab "Parameters");
- presenting a quick overview of successor relationships of the selected component (tab "Successors") – thus complementing the display of the whole model above;
- a detailed and flexible examination of simulation results referring to a single-row or multiple-row component selection in the tree view (tab "Evaluation", cmp. figure 7).

Strictly following an object-oriented design, both server and client have been programmed in Java. For the latter the reasons are obvious: It allows the client to be run as an applet. Thus, no software installation is required on computers where the client is to be run – and it remains platform-independent. Besides this independence, the enormous variety of highly useful third-party libraries was a strong argument for choosing Java as programming language for the server as well. Our application heavily relies on such libraries, e. g. JDOM for XML processing, JFreeChart for graphical visualization of simulation results (see section "Integrating the Client with Web-based Learning Modules: An Example" below), and several libraries from the Apache Software Foundation.

The communication between client and server is very straightforward. Whenever the server is contacted by a client, its request can be answered with a single reply:

- the initial load request will be answered by providing the model structure and its initial parameterization;
- a user-initiated request for (re)loading a previously saved parameterization (and simulation results, if available) will be answered by providing the desired data;
- transferring the current parameterization together with a save request will cause the server to store the data transmitted;
- starting the simulation based on the current parameterization of the model will block the client until it receives the simulation results from the server.

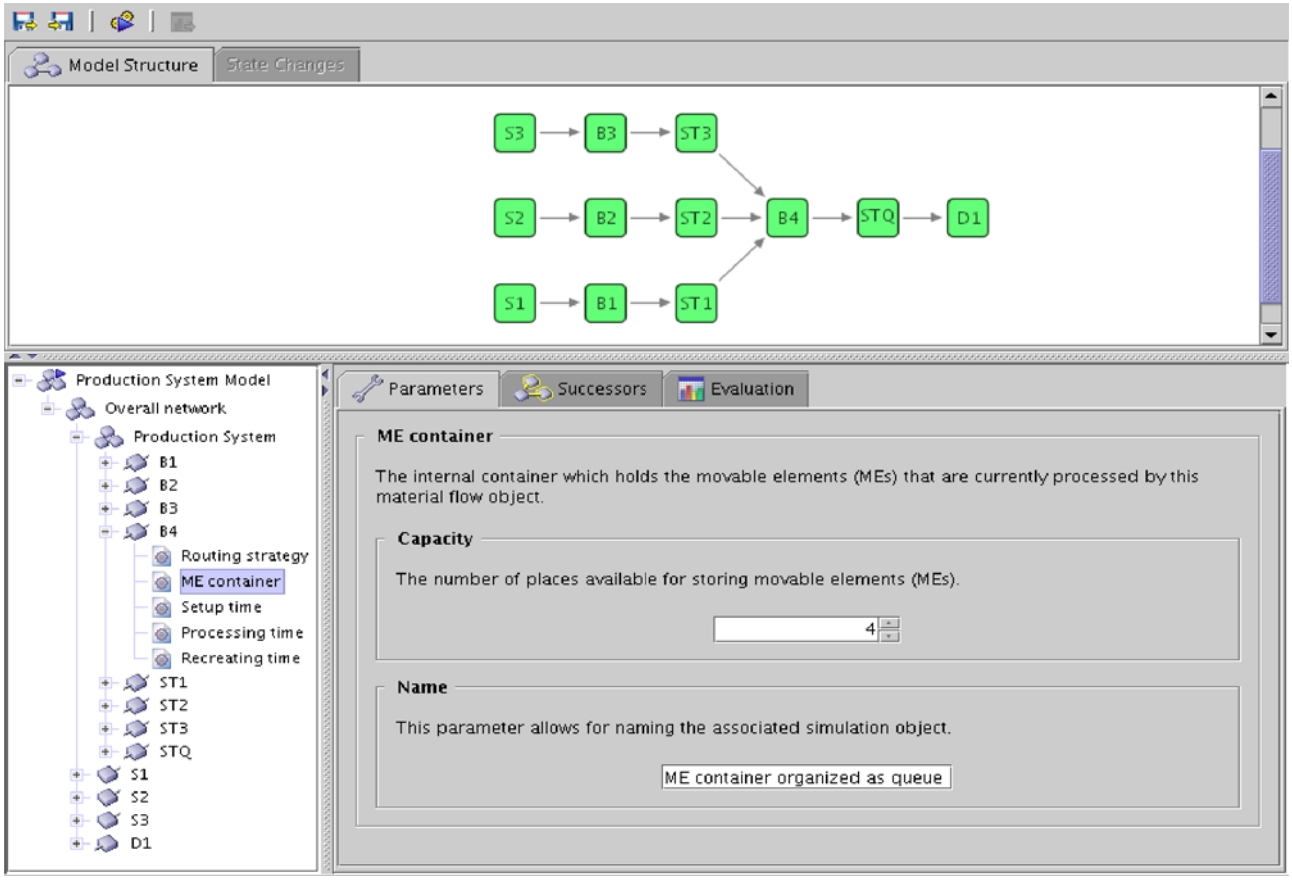

Figure 4: The Client Interface

Thus, implementing client requests as remote procedure calls looks reasonable. For the sake of simplicity, the prototype employs the XML-RPC protocol, although a replacement by Java's RMI is likely to follow in the future.

INTEGRATING THE CLIENT WITH WEB-BASED LEARNING MODULES: AN EXAMPLE

To demonstrate some of the main features of the client application, a typical simulation-related task from a training module belonging to the master-degree course "System Optimization / Simulation Techniques" shall be considered. Given a simple production system consisting of three single-processor workstations that direct their output to a quality assessment station with an input buffer of very limited capacity, it is assumed that the quality assessment is done manually by a worker who stops working from time to time (in order to smoke a cigarette, have lunch etc.) whereas the production within the three single-processor work-stations is fully automated. It is clear that, whenever the QA worker takes a break, the queue of jobs waiting in the buffer will lengthen until the worker is back at his place to continue his work, thus emptying the buffer after some time. Eventually, the job queue will exhaust the capacity of the buffer, thus blocking the production within the preceding single-processor workstations.

The initial parameterization of the simulation model will regularly lead to such interruptions of the production, due to an insufficient capacity of the buffer preceding the QA station. Bearing in mind that each additional place in the buffer will cost a certain amount of money while for each of the three single-processor workstations a production stop causes a certain loss of return, the task is to resize the buffer in such a way that interruptions of production at the workstations are reduced to a minimum while the costs for enlarging the buffer are kept as low as possible.

Having started the client, the user will probably run the simulation prior to any modifications of the parameterization, in order to get a first idea of the dynamic system behaviour. As already pointed out, the simulation server returns a variety of results to the client that can be investigated by selecting workflow components of specific interest in the tree view and then evaluating the tables and charts available below the "Evaluation" tab. Figures 5-7 illustrate some of these evaluation possibilities:

- Figure 5 shows the ratio of operating states for the three single-processor workstations of the simulation model. The cumulative visualization in a single chart is achieved by selecting all three workstations in the tree view of the client interface (multiple-row selection). Referring to the example scenario, the chart illustrates the adverse relation of machine down time (operating state "blocked") to times compassed for other operating states.
- Figure 6 depicts a chart that visualizes changes of the operating state of components over simulation time. As an example, the QA station and one workstation have been selected in the tree view. The resulting chart clearly shows the immediate effect that a pause of the QA worker (denoted by operating state "failure") has on production.
- As a countercheck, the user may want to examine the changes of content of the input buffer of the QA station over simulation time. A table view of the content log of this buffer is depicted in figure 7.

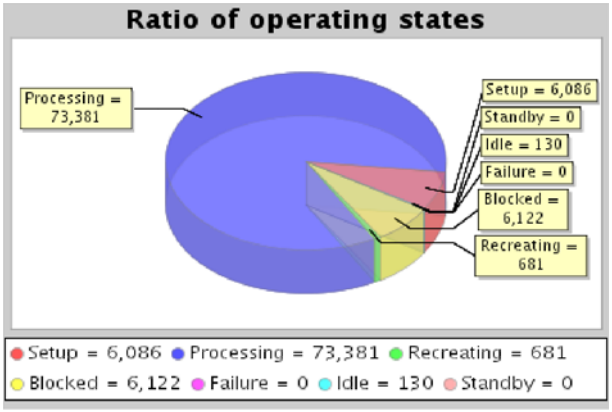

Figure 5: Example Chart Showing Ratio of Operating States of Selected Components

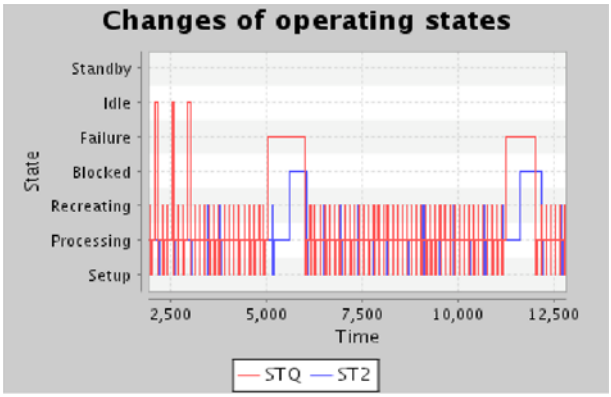

Figure 6: Example Chart Showing Changes of Operating State Over Time

As figure 5 suggests, the buffer B4 preceding the QA station is indeed a bottleneck in the system. So a first step towards improving system dynamics such that the requirements as specified in the task are met would be an attemptive increase of B4's capacity. This can be achieved on the tab "Parameters" as showed in figure 4. After the modified parameterization has been saved, a

subsequent simulation run can be performed that will evolve the effects of the changes. The achievement of a real solution of the task will require further calculations regarding the cost aspects mentioned above. As randomness plays an important role with respect to modelling temporal behaviour of components by means of probability distributions (Law and Kelton 2000), repeated simulation runs based on the same parameterization will prove validity of the solution.

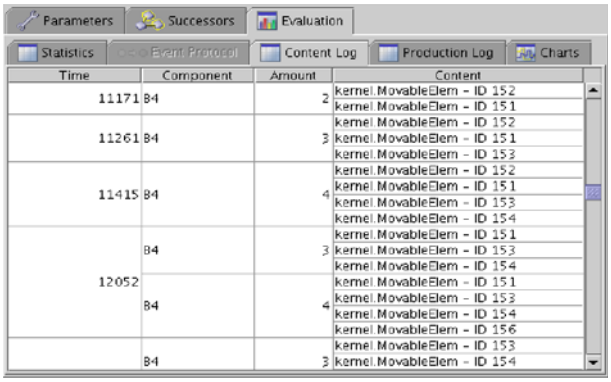

Figure 7: Table View of Content Log of a Buffer Component

CONCLUSION

This paper presents an approach to blended learning which combines interactive simulation-based training modules with web-based learning units and traditional lectures. The concept outlined before has been employed successfully in the master-degree course "System Optimization / Simulation Techniques" at the University of Wuppertal. As described in the preceding example, the concept allows for experimenting with the parameters of a given production system to figure out their impact on the system's dynamic behaviour. The benefits for students are obvious, as they are able to gain experience in this area, thus complementing the knowledge of the theoretical background acquired during lectures and via e-learning.

As the system here presented is still at a preliminary stage, there is plenty of room for enhancements. Further research will include

- an even wider variety of possibilities for examining and visualizing simulation results;
- an optimization of the communication between server and clients;
- a closer integration of the graphical client with the learning management system ILIAS;
- improvement and further development of contents and structure of the course "System Optimization / Simulation Techniques" based on the experience collected during the pilot installation of the prototype simulation application.

Having reached a higher degree of maturity, the system could be adopted to industrial use. Deployed in the field of internal training, it could deliver fundamental modelling experience to planners who need to become familiar with the ideas and concepts of production simulation in the run-up of a simulation project.

REFERENCES

Banks, J.; J.S. Carson; B.L. Nelson; and D.M. Nicol. 2005. *Discrete-Event System Simulation*. Prentice Hall.

Becker, B.-D. 1991. *Simulationssystem für Fertigungsprozesse mit Stückgutcharakter*. Springer-Verlag.

Kühn, W. 2002. "Java-Sim – An Advanced Discrete-Event Simulation Library". In *Proceedings of the 2002 Summer Computer Simulation Conference* (San Diego, CA).

Kühn, W. 2005. "Integration von Discrete-Event Simulation mit der Open Source Lernplattform ILIAS". In *ILIAS Conference Reader* (Nürnberg, Germany, 34-47).

Law, A.M. and W.D. Kelton. 2000. *Simulation Modeling and Analysis, 3/e*. McGraw-Hill.

Wiedemann, T. 2002. "Next Generation Simulation Environments Founded on Open Source Software and XML-Based Standard Interfaces". In *Proceedings of the 2002 Summer Computer Simulation Conference* (San Diego, CA, 623-628).

AUTHOR BIOGRAPHIES

WOLFGANG KÜHN studied mechanical engineering at the University of Brunswick, Germany. Afterwards he worked with Blaupunkt for two years. At the University of Bremen he got his PhD in production engineering in 1991, and his habilitation in the area of simulation of production systems in 1997. From 1993 to 1995 he worked as Associate Professor at the Asian Institute of Technology in Bangkok. In 1996 he founded the SIPOC Simulation based Planning, Optimization and Control GmbH in Bremen. Since 1997 he has been working as a professor for production planning and control in the department of Electrical, Information and Media Engineering at the University of Wuppertal.

MICHAEL KORDT studied computer science at the University of Kaiserslautern, Germany. Having graduated in early 2001, he worked with Robert Bosch GmbH, Stuttgart, for two and a half years. Since late 2003 he has been working as a scientific assistant at the chair for production planning and control in the department of Electrical, Information and Media Engineering at the University of Wuppertal.

ROLAND GRAH is a student of mathematics and computer science at the University of Wuppertal. Since 2004 he has been involved in the project here presented.